The International Workshop
on
Coding and Cryptography
(WCC 13)

April 15–19, 2013
Bergen, Norway

PREPROCEEDINGS
PREPROCEEDINGS

The International Workshop on Coding and Cryptography
WCC 2013
April 15–19, 2013
Bergen (Norway)

Organized by the Selmer Center at the University of Bergen, Norway and INRIA, Rocquencourt, France.

Editors: Lilya Budaghyan, Tor Helleseth, Matthew G. Parker

Bergen 2013
Preface

This volume contains the pre-proceedings of the 8-th International Workshop on Coding and Cryptography held in Bergen, Norway, April 15-19, 2013. This series of workshops have been organized, biannually, since 1999. The previous workshops have alternated between Paris and Bergen (and their beautiful surroundings).

This year the workshop received 133 submissions in total, this being a new record for these workshops. The program committee therefore had a particularly challenging task to select the 54 papers to be presented at the conference. A special feature this year was that the European project COST 1104: Random Network Coding and Designs over GF(q) integrated their workshop with WCC 2013. This choice was made because many researchers involved with COST 1104 were also on the TPC of WCC and/or active at previous workshops. COST submissions were refereed as regular WCC submissions and accepted COST papers will be presented during the last two days of the workshop.

We are very pleased to have invited talks by Kai-Uwe Schmidt (Otto-von-Guericke Universität, Magdeburg, Germany), Gregor Leander (Ruhr-Universität, Bochum, Germany) and Frank Kschischang (University of Toronto, Canada).

The authors of the papers in these pre-proceedings will be invited to submit a full paper, based on their WCC submission, to a special invited edition of Designs, Codes and Cryptography. These papers will be refereed again to ensure that they meet the high standard of this journal. We hope the conference and resulting discussion will inspire submissions to this special issue.

A special thanks goes to the General Co-Chairs Øyvind Ytrehus and Alexander Kholosha and to Oleksandr Kazymyrov for handling the web-based review system. We gratefully acknowledge the generous financial support from the Norwegian Research Council and from Bergen Universitetsfond.

We wish you all a pleasant stay in Bergen and an inspiring workshop.

April 2013,

Lilya Budaghyan, Tor Helleseth, Matthew G. Parker
Program Committee

Daniel Augot (GRACE, INRIA, Saclay - Ile de France, France)
Angela Barbero (University of Valladolid, Spain)
Andrey Bogdanov (Technical University of Denmark, Lyngby)
Lilya Budaghyan (co-Chair, University of Bergen, Norway, and Universities of Paris 8 and 13, France)
Anne Canteaut (INRIA, Rocquencourt, France)
Claude Carlet (University of Paris 8, France)
Joan Daemen (STMicroelectronics, Switzerland)
Cunsheng Ding (The Hong Kong University of Science and Technology, China)
Lars Eirik Danielsen (Webstep, Norway)
Olav Geil (Aalborg University, Denmark)
Guang Gong (University of Waterloo, Canada)
Marcus Greferath (University College Dublin, Ireland)
Tor Helleseth (co-Chair, University of Bergen, Norway)
Camilla Hollanti (Aalto University, Finland)
Jonathan Jedwab (Simon Fraser University, Canada)
P. Vijay Kumar (Indian Institute of Science, India)
Gary McGuire (University College Dublin, Ireland)
Wilfried Meidl (Sabanci University, Turkey)
Sihem Mesnager (University of Paris 8, France)
Kaisa Nyberg (Aalto University, Finland)
Patric Östergård (Aalto University, Finland)
Matthew G. Parker (co-Chair, University of Bergen, Norway)
Alexander Pott (University of Magdeburg, Germany)
Bart Preneel (Katholieke Universiteit Leuven, Belgium)
Vincent Rijmen (Katholieke Universiteit Leuven, Belgium)
Eirik Rosnes (Ceragon, Norway)
Igor Semaev (University of Bergen, Norway)
Sondre Rønjøm (NSM, Norway)
Nicolas Sendrier (INRIA, Rocquencourt, France)
Xiaohu Tang (Southwest Jiaotong University, China)
Arne Winterhof (Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austria)
Kyeongcheol Yang (Postech, Pohang, S. Korea)
Øyvind Ytrehus (University of Bergen, Norway)
Victor Zinoviev (IPIT, Moscow, Russia)

General Co-chairs

Alexander Kholosha (co-Chair, University of Bergen, Norway)
Øyvind Ytrehus (co-Chair, University of Bergen, Norway)
Table of Contents

Preface iii

Table of Contents, Program v

Invited Talk 1
Kai-Uwe Schmidt, “Low autocorrelation sequences” 15

Sequences
Elena Dubrova, “A method for generating full cycles by a composition of NLFSRs” 16
Gottlieb Pirsic and Arne Winterhof, “On discrete Fourier transform, ambiguity, and Hamming-autocorrelation of pseudorandom sequences” 30

APN Functions
Florian Caullery, “APN functions of degree $4e$ with $e \equiv 3 \pmod{4}$” 39
Yuyin Yu, Mingsheng Wang and Yongqiang Li, “A matrix approach for constructing quadratic APN functions” 47
Guobiao Weng, Yin Tan and Guang Gong, “On quadratic almost perfect nonlinear functions and their related algebraic object” 57

Coding Theory
Torleiv Kløve and Moshe Schwartz, “Covering sets for limited-magnitude errors” 69
Nicolas Sendrier and Dimitris E. Simos, “How easy is code equivalence over $F_q$?” 79
Sarah E. Anderson and Gretchen L. Matthews, “Exponents of polar codes using algebraic geometric code kernels” 92
Hyun Kwang Kim and Phan Thanh Toan, “New inequalities for q-ary constant-weight codes” 102

Feistel Ciphers
Kyoji Shibutani and Andrey Bogdanov, “Towards the optimality of Feistel ciphers with substitution-permutation functions” 112
Shingo Yanagihara and Tetsu Iwata, “Type 1.x generalized Feistel structures” 124
Mahabir Prasad Jhanwar, Ayineedi Venkateswarlu and Reihaneh Safavi-Naini, “Paillier-based publicly verifiable (non-interactive) secret sharing” 135
Quantum Algorithms
Kaushik Chakraborty and Subhamoy Maitra, “Quantum algorithm to check resiliency of a Boolean function (Extended Abstract)” 150
“Petr Lisoněk and Vijaykumar Singh, Construction X for quantum errorcorrecting codes” 159

Invited Talk 2
Gregor Leander, “Design and analysis of block ciphers - Links to Boolean functions and coding theory” 166

Coding and Cryptography
Philippe Gaborit, Gaëtan Murat, Olivier Ruatta and Gilles Zémor, “Low rank parity check codes and their application to cryptography” 167
Alain Couvreur, Philippe Gaborit, Valrie Gauthier, Ayoub Otmani and Jean-Pierre Tillich, “Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes” 180

Cryptanalysis
Maria Eichlseder, Florian Mendel, Tomislav Nad, Vincent Rijmen and Martin Schläffer, “Linear propagation in efficient guess-and-determine attacks” 193
Thomas Johansson and Carl Löndahl, “A new algorithm for finding low-weight polynomial multiples and its applications to TCHo” 203
Santanu Sarkar, “Small secret exponent attack on RSA variant with modulus $N = p^2q$” 215

Finite Fields
Céline Blondeau and Léo Perrin, “More differentially 6-uniform power functions” 223
Kai-Uwe Schmidt and Yue Zhou, “Planar functions over fields of characteristic two (Extended Abstract)” 234
Gohar Kyureghyan and Qi Wang, “An upper bound on the size of Kakeya sets in finite vector spaces” 243
Grasiele C. Jorge, Antonio Campello and Sueli R. Costa, “$q$-ary lattices in the lp norm and a generalization of the Lee metric” 250

Algebraic Coding Theory
Stéphane Ballet and Robert Rolland, “On low weight codewords of generalized affine and projective Reed-Muller codes (Extended Abstract)” 258
Morgan Barbier, Clément Pernet and Guillaume Quintin, “On the decoding of quasi-BCH codes” 268
Irene Platoni, “Quasi-perfect linear codes from plane cubics” 278
Optical Orthogonal Codes

Jin-Ho Chung and Kyeongcheol Yang, “A new class of optimal variable-weight optical orthogonal codes” 285
Kenneth Shum, “Optimal three-dimensional optical orthogonal codes and related combinatorial designs” 294

Cryptography

Santanu Sarkar, “Further non-randomness in RC4, RC4A and VMPC” 304
Vincent Rijmen, Deniz Toz and Kerem Varici, “On the four round AES characteristics” 315
Hadi Soleimany and Kaisa Nyberg, “Zero-correlation linear cryptanalysis of reduced-round Lblock” 329
Felix Fontein, Michael Schneider and Urs Wagner, A polynomial time version of LLL with deep insertions” 343

Boolean Functions

Ayça Çeşmelioğlu, Wilfried Meidl and Alexander Pott, “On the normality of pary bent functions” 355
Claude Carlet and Andrew Klapper, “On the arithmetic Walsh coefficients of Boolean functions” 363
Subhabrata Samajder and Palash Sarkar, “Fast multiplication of the algebraic normal forms of two Boolean functions” 373

Finite Fields and Rings

Thomas Feulner, “On canonical forms of ring-linear codes” 385
Faruk Göloğlu and Gary McGuire, “When is $x - 1 + L(x)$ a permutation in odd characteristic?” 397
Ferruh Özbudak and Zülfükar Saygı, “On the exact number of solutions of certain linearized equations.” 407
Arpita Maitra and Goutam Paul, “Symmetric incoherent eavesdropping against MDI QKD” 413

Lattices and Codes

Wittawat Kositwattanarerk and Frédérique Oggier, “On construction D and related constructions of lattices from linear codes” 427
Soon Sheng Ong and Frédérique Oggier, “Lattices from totally real number fields with large regulator” 437
David A. Karpuk, Camilla Hollanti and Emanuele Viterbo, “Probability bounds for two-dimensional algebraic lattice codes” 447
Invited Talk 3
Frank Kschischang, “An algebraic approach to physical-layer network coding” 458

Network Coding and Geometry
Ivan Landjev and Peter Vandendriesche, “On the rank of incidence matrices in projective Hjelmslev spaces” 459
Linda Beukemann, Klaus Metsch and Leo Storme, “On weighted \((\delta \nu_{\mu+1}, \delta \nu_{\mu}; k - 1, q)\)-minihypers, \(q\) square” 466

Network Coding and Related Topics 1
Eimear Byrne, “On bounds for network codes” 475
Heide Gluesing-Luerssen, “Tail-biting trellis realizations and local reductions” 485
Iván Blanco Chacón, Dionis Remón and Camilla Hollanti, “Fuchsian codes for AWGN channels” 495
Jinquan Luo, “Weight distribution of cyclic codes with several non-zeroes” 508

Gabidulin Codes
Wenhui Li, Vladimir Sidorenko and Di Chen, “On transform-domain decoding of Gabidulin codes” 516
Antonia Wachter-Zeh and Alexander Zeh, “Interpolation-based decoding of interleaved Gabidulin codes” 527
Anna-Lena Trautmann, Natalia Silberstein and Joachim Rosenthal, “List decoding of Lifted Gabidulin codes via the Plücker embedding” 538

Network Coding and Related Topics 2
Michael Braun, Tuvi Etzion, Patric R. J. Östergård, Alexander Vardy and Alfred Wassermann, “On the existence of \(q\)-analogs of Steiner systems” 549
Johan S. R. Nielsen and Alexander Zeh, “Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes” 557
Emanuele Bellini, Eleonora Guerrini and Massimiliano Sala, “Some bounds on the size of codes” 568
Program WCC 2013

Monday, April 15

Invited Talk 1
Chairman: Tor Helleseth
9:00 – 9:50 Kai-Uwe Schmidt, Low autocorrelation sequences

Sequences
9:50 – 10:15 Elena Dubrova, A method for generating full cycles by a composition of NLFSRs
10:15 – 10:40 Gottlieb Pirsic and Arne Winterhof, On discrete Fourier transform, ambiguity, and Hamming-autocorrelation of pseudorandom sequences
10:40 – 11:10 Coffee Break

APN Functions
Chairman: Claude Carlet
11:10 – 11:35 Florian Caullery, APN functions of degree 4e with e ≡ 3 (mod 4)
11:35 – 12:00 Yuyin Yu, Mingsheng Wang and Yongqiang Li, A matrix approach for constructing quadratic APN functions
12:00 – 12:25 Guobiao Weng, Yin Tan and Guang Gong, On quadratic almost perfect nonlinear functions and their related algebraic object
12:30 – 14:00 Lunch

Coding Theory
Chairman: Daniel Augot
14:00 – 14:25 Torleiv Kløve and Moshe Schwartz, Covering sets for limited-magnitude errors
14:25 – 14:50 Nicolas Sendrier and Dimitris E. Simos, How easy is code equivalence over \( F_q \)?
14:50 – 15:15 Sarah E. Anderson and Gretchen L. Matthews, Exponents of polar codes using algebraic geometric code kernels
15:15 – 15:40 Hyun Kwang Kim and Phan Thanh Toan, New inequalities for q-ary constant-weight codes
15:40 – 16:10 Coffee Break

Feistel Ciphers
Chairman: Anne Canteaut
16:10 – 16:35 Kyoji Shibutani and Andrey Bogdanov, Towards the optimality of Feistel ciphers with substitution-permutation functions
16:35 – 17:00 Shingo Yanagihara and Tetsu Iwata, Type I.x generalized Feistel structures
17:00 – 17:25 Mahabir Prasad Jhanwar, Ayineedi Venkateswarlu and Reihaneh Safavi-Naini, Paillier-based publicly verifiable (non-interactive) secret sharing

Quantum Algorithms
Chairman: Matthew Parker
17:25 – 17:50 Kaushik Chakraborty and Subhamoy Maitra, Quantum algorithm to check resiliency of a Boolean function (Extended Abstract)
17:50 – 18:15 Petr Lisoněk and Vijaykumar Singh, Construction X for quantum error-correcting codes
Tuesday, April 16

Invited Talk 2
Chairman: Vincent Rijmen
9:00 – 9:50 Gregor Leander, Design and analysis of block ciphers - Links to Boolean functions and coding theory

Coding and Cryptography
9:50 – 10:15 Philippe Gaborit, Gaëtan Murat, Olivier Ruatta and Gilles Zémor, Low rank parity check codes and their application to cryptography
10:15 – 10:40 Alain Couvreur, Philippe Gaborit, Valérie Gauthier, Ayoub Otmani and Jean-Pierre Tillich, Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes

10:40 – 11:10 Coffee break

Cryptanalysis
Chairman: Andrey Bogdanov
11:10 – 11:35 Maria Eichlseder, Florian Mendel, Tomislav Nad, Vincent Rijmen and Martin Schläffer, Linear propagation in efficient guess-and-determine attacks
11:35 – 12:00 Thomas Johansson and Carl Lönndahl, A new algorithm for finding low-weight polynomial multiples and its applications to TCHo
12:00 – 12:25 Santanu Sarkar, Small secret exponent attack on RSA variant with modulus $N=p^2q$

12:30 – 14:00 Lunch

Finite Fields
Chairman: Alexander Pott
14:00 – 14:25 Céline Blondeau and Léo Perrin, More differentially 6-uniform power functions
14:25 – 14:50 Kai-Uwe Schmidt and Yue Zhou, Planar functions over fields of characteristic two (Extended Abstract)
14:50 – 15:15 Gohar Kyureghyan and Qi Wang, An upper bound on the size of Kakeya sets in finite vector spaces
15:15 – 15:40 Grasiele C. Jorge, Antonio Campello and Sueli R. Costa, $q$-ary lattices in the $l_p$ norm and a generalization of the Lee metric

15:40 – 16:10 Coffee Break

Algebraic Coding Theory
Chairman: Nicolas Sendrier
16:10 – 16:35 Stéphane Ballet and Robert Rolland, On low weight codewords of generalized affine and projective Reed-Muller codes (Extended Abstract)
16:35 – 17:00 Morgan Barbier, Clément Pernet and Guillaume Quintin, On the decoding of quasi-BCH codes
17:00 – 17:25 Irene Platoni, Quasi-perfect linear codes from plane cubics

Optical Orthogonal Codes
Chairman: Alexander Kholosha
17:25 – 17:50 Jin-Ho Chung and Kyeongcheol Yang, A new class of optimal variable-weight optical orthogonal codes
17:50 – 18:15 Kenneth Shum, Optimal three-dimensional optical orthogonal codes and related combinatorial designs
**Wednesday, April 17**

**Cryptography**  
*Chairman: Thomas Johansson*  
9:00 – 9:25 Santanu Sarkar, *Further non-randomness in RC4, RC4A and VMPC*  
9:25 – 9:50 Vincent Rijmen, Deniz Toz and Kerem Varıcı, *On the four round AES characteristics*  
9:50 – 10:15 Hadi Soleimany and Kaisa Nyberg, *Zero-correlation linear cryptanalysis of reduced-round Lblock*  
10:15 – 10:40 Felix Fontein, Michael Schneider and Urs Wagner, *A polynomial time version of LLL with deep insertions*  
10:40 – 11:10 *Coffee Break*  

**Boolean Functions**  
*Chairman: Sihem Mesnager*  
11:10 – 11:35 Ayça Çesmioğlu, Wilfried Meidl and Alexander Pott, *On the normality of p-ary bent functions*  
11:35 – 12:00 Claude Carlet and Andrew Klapper, *On the arithmetic Walsh coefficients of Boolean functions*  
12:00 – 12:25 Subhabrata Samajder and Palash Sarkar, *Fast multiplication of the algebraic normal forms of two Boolean functions*  
12:30 – 14:00 *Lunch*  
15:00 – 18:00 *Excursion*
Thursday, April 18

Finite Fields and Rings
Chairman: Pascale Charpin
9:00 – 9:25 Thomas Feulner, *On canonical forms of ring-linear codes*
9:25 – 9:50 Faruk Göloğlu and Gary McGuire, *When is \( x^{-1} + L(x) \) a permutation in odd characteristic?*
9:50 – 10:15 Ferruh Özbudak and Zülfükar Saygı, *On the exact number of solutions of certain linearized equations*
10:15 – 10:40 Arpita Maitra and Goutam Paul, *Symmetric incoherent eavesdropping against MDI QKD*

10.40 – 11.10 Coffee break

Lattices and Codes
Chairman: Patric R. J. Östergård
11:10 – 11:35 Wittawat Kositwattanarerk and Frédérique Oggier, *On construction D and related constructions of lattices from linear codes*
11:35 – 12:00 Soon Sheng Ong and Frédérique Oggier, *Lattices from totally real number fields with large regulator*
12:00 – 12:25 David A. Karpuk, Camilla Hollanti and Emanuele Viterbo, *Probability bounds for two-dimensional algebraic lattice codes*

12.30 – 14:00 Lunch

Invited Talk 3
Chairman: Marcus Greferath
14:00 – 14:50 Frank Kschischang, *An algebraic approach to physical-layer network coding*

Network Coding and Geometry
14:50 – 15:15 Ivan Landjev and Peter Vandendriesche, *On the rank of incidence matrices in projective Hjelmslev spaces*
15:15 – 15:40 Linda Beukemann, Klaus Metsch and Leo Storme, *On weighted \((\delta v_{\mu,1}, \delta v_{\mu}; k-1,q)\)-minihypers, \( q \) square*

15.40 – 16:10 Coffee break

Network Coding and Related Topics 1
Chairman: Camilla Hollanti
16:10 – 16:35 Eimear Byrne, *On bounds for network codes*
16:35 – 17:00 Heide Gluesing-Luerssen, *Tail-biting trellis realizations and local reductions*
17:00 – 17:25 Iván Blanco Chacón, Dionis Remón and Camilla Hollanti, *Fuchsian codes for AWGN channels*
17:25 – 17:50 Jinquan Luo, *Weight distribution of cyclic codes with several non-zeroes*

19:00 Banquet
Friday, April 19

Gabidulin Codes
Chairman: Tuvi Etzion
9:00 – 9:25 Wenhui Li, Vladimir Sidorenko and Di Chen, *On transform-domain decoding of Gabidulin codes*
9:50 – 10:15 Anna-Lena Trautmann, Natalia Silberstein and Joachim Rosenthal, *List decoding of Lifted Gabidulin codes via the Plücker embedding*

10:15 – 10:45 Coffee break

Network Coding and Related Topics 2
Chairman: Simon Blackburn
10:45 – 11:10 Michael Braun, Tuvi Etzion, Patric R. J. Östergård, Alexander Vardy and Alfred Wassermann, *On the existence of q-analogs of Steiner systems*
11:10 – 11:35 Johan S. R. Nielsen and Alexander Zeh, *Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes*
11:35 – 12:00 Emanuele Bellini, Eleonora Guerrini and Massimiliano Sala, *Some bounds on the size of codes*

12:00 – 13:00 Lunch

END of WORKSHOP

13:00 – 17:00 COST - Management Committee (MC) – meeting
Kai-Uwe Schmidt
Faculty of Mathematics, Otto-von-Guericke University,
Magdeburg, Germany

Low autocorrelation sequences

Abstract:
The extent to which a sequence of finite length differs from shifted versions of itself is measured by its aperiodic autocorrelations. The identification of binary sequences with collectively small aperiodic autocorrelations is an old problem in combinatorial optimisation and complex analysis, with practical importance in engineering and physics. I shall describe recent progress in answering the two central questions in this area: How small can the aperiodic autocorrelations of a binary sequence collectively be and how can we efficiently find the best sequences? The methods are of probabilistic, combinatorial, and analytic flavour and lead to new records and to resolutions of several conjectures.
A Method for Generating Full Cycles by a Composition of NLFSRs

Elena Dubrova

Abstract Non-Linear Feedback Shift Registers (NLFSR) are a generalization of Linear Feedback Shift Registers (LFSRs) in which a current state is a non-linear function of the previous state. The interest in NLFSRs is mostly motivated by their ability to generate pseudo-random sequences which are usually hard to break with existing cryptanalytic methods. However, it is still not known how to construct large $n$-stage NLFSRs which generate full cycles of $2^n$ possible states. This paper presents a method for generating full cycles by a composition of NLFSRs. First, we show that an $n \times k$-stage register with period $O(2^{2n})$ can be constructed from $k$ $n$-stage NLFSRs by adding to their feedback functions a logic block of size $O(n \times k)$, for $k > 1$. This logic block implements Boolean functions representing the set of pairs of states whose successors have to be exchanged in order to join cycles. Then, we show how to join all cycles into one by using one more logic block of size $O(n \times k^2)$ and an extra time step.

Keywords LFSR · NLFSR · full cycle · de Bruijn sequence · composition

1 Introduction

Non-Linear Feedback Shift Registers (NLFSR) are a generalization of Linear Feedback Shift Registers (LFSRs) in which a current state is a non-linear function of the previous state [18]. The interest in NLFSRs is motivated to a large extent by their ability to generate pseudo-random sequences which are usually hard to break with existing cryptanalytic methods [23]. While LFSRs are widely used in testing and simulation [6], for cryptographic applications their pseudo-random sequences are not secure. The structure of an $n$-bit LFSR can be easily deduced from $2n$ consecutive bit of its sequence by using the Berlekamp-Massey algorithm [21]. Contrary, an adversary might need up to $O(2^n)$ bits of a sequence to determine the structure of the $n$-bit NLFSR which generates it [10].
However, while the theory behind LFSRs is well-understood, many fundamental problems related to NLFSRs remain open. One of the most important ones is constructing an NLFSR with the maximum period. It is known that an \( n \)-stage LFSR has the maximum period of \( 2^n - 1 \) if and only if its characteristic polynomial is primitive \([18]\). For NLFSRs, no similar property has been found so far. Small NLFSRs with the maximum period can be computed by simulation. However, with today’s processing power, we can simulate NLFSRs of size \( n < 35 \) only \([7]\). This is not enough for cryptographic applications, which require periods larger that \( 2^{128} \) \([22]\).

Sequences generated by maximum-period NLFSRs are also known as de Bruijn sequences. In a de Bruijn sequence of order \( n \) all \( 2^n \) different binary \( n \)-tuples appear exactly once. It is known that the number of different de Bruijn sequences of order \( n \) is \( 2^{2^n - 1} \) \([3]\).

An excellent survey of algorithms for generating de Bruijn sequences using shift registers is given in \([13]\). These algorithms can be classified into two groups.

Algorithms in the first group start from a shift register producing several shorter cycles and then join them into one cycle. Fredricksen \([12]\) have shown how to generate full cycles from a circulating register of length \( n \) using \( 6n \) bits of storage and \( n \) time steps to produce the next state from a current state. The algorithm of Etzion and Lempel \([11]\) generates full cycles from a pure summing register using \( n^2/4 \) bits of storage and \( n \) time steps to produce the next state. Jansen \([20]\) presented an algorithm for joining state cycles of an arbitrary shift register. This algorithm generates full cycles using \( 3n \) bits of storage and at most \( 4n \) time steps for producing the next state from a current state.

A recursive algorithm for generating de Bruijn sequences based on Lempel’s D-homomorphism was presented by Annexstein \([1]\). A non-recursive version of this algorithm was proposed by Chang et al in \([4]\). In both algorithms, \( n \)-variable Boolean functions generating de Bruijn sequence of order \( n \) are constructed from Boolean functions with a smaller number of variables.

Algorithms in the second group start from a shift register with a known period and obtain another shift register with the same period by using cross-join pairs. Different approaches to selecting cross-join pairs have been explored. Fredricksen \([14]\) proposed an algorithm for a class of NLFSRs generating “prefer one” de Bruijn sequences. In a “prefer one” de Bruijn sequence, the \( n \)-tuple \((1, a_1, a_2, \ldots, a_{n-1})\) precedes the \( n \)-tuple \((0, a_1, a_2, \ldots, a_{n-1})\) for all \( n - 1 \)-tuples \((a_1, a_2, \ldots, a_{n-1})\) except all-0. Dubrova \([8]\) presented a method for constructing NLFSRs with period \( 2^n - 1 \) from maximum-period LFSRs in which the cross-join pairs are determined by a non-linear function added to the LFSR. The number of possible cross-join pairs in maximum-period LFSRs has been derived by Helleseth and Klove \([19]\).

This paper presents a method for generating full cycles by a composition of NLFSRs. First, we show how to construct \( n + k \)-stage register with period \( O(2^{2n}) \) from \( k \) \( n \)-stage NLFSRs, for \( k > 1 \). We derive Boolean functions representing the set of pairs of states whose successors have to be exchanged in order to join cycles. These functions are added to the feedback functions of \( n \)-stage NLFSRs. We prove that the additional logic can be implemented using \( O(n + k) \) 2-input gates. Then, we show how to join all cycles into one by using one more logic block of size \( O(n + k^2) \) and an extra time step.

In a full cycle generated by a traditional NLFSR, each current and next state overlap in all but one positions. In \( n \times k \)-stage registers constructed using the presented method, each current and next state overlap in all but \( k \) positions. Therefore, their full cycles differ from the full cycles generated by NLFSRs and their output sequences are not of de Bruijn type.

The paper is organized as follows. Section 2 gives a background on NLFSRs. Section 3 introduces \((n, k)\)-composed registers. Section 4 analyses properties of \((n, k)\)-composed registers. Section 5 presents an algorithm for constructing registers with period \( O(2^{2n}) \) from...
\( (n, k) \)-composed registers. Section 6 presents an algorithm which joins all cycles into one. Section 7 concludes the paper.

2 Preliminaries

Throughout the paper, we use "\oplus\) and "\cdot\) to denote addition and multiplication in \( GF(2) \), respectively, and "\(+\) and "\(\times\)" to denote arithmetic addition and multiplication, respectively. We use \( \pi \) to denote the complement of \( x \), defined by \( \pi = 1 \oplus x \).

An \( n \)-stage Non-Linear Feedback Shift Register (NLFSR) consists of \( n \) binary storage elements, called stages [18]. Each stage \( i \in \{1, 2, \ldots, n\} \) has an associated state variable \( x_i \in \{0, 1\} \) which represents the current value of the stage. At each clock cycle, the value of \( x_{i+1} \) is transferred to \( x_i \), for all \( i \in \{1, 2, \ldots, n-1\} \). The feedback function \( f(x_1, x_2, \ldots, x_n) \), computed from the content of the \( n \) stages, determines the next value of \( x_n \). The output of an NLFSR is the sequence of bits appearing in its stage 1.

The feedback function \( f \) induces the mapping \( F : \{0, 1\}^n \to \{0, 1\}^n \) of type
\[
(x_1, x_2, \ldots, x_n) \mapsto (x_2, x_3, \ldots, x_n, f(x_1, x_2, \ldots, x_n)).
\]

The state of an \( n \)-stage register is a vector of values \( S = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n \) of its state variables \( x_1, x_2, \ldots, x_n \).

A cycle of length \( m \) of an \( n \)-stage register is a vector of states \( (S_0, S_1, \ldots, S_{m-1}) \) such that \( F(S_i) = S_{i+1} \), for \( i \in \{0, 1, \ldots, m-2\} \), and \( F(S_{m-1}) = S_0 \). The period of a register is the length of its longest cycle\(^1\).

A necessary and sufficient condition for an NLFSR to be branchless [18] is that its feedback function \( f \) can be written in the form
\[
f(x_1, x_2, \ldots, x_n) = x_1 \oplus g(x_2, \ldots, x_n),
\]
where \( g \) is a Boolean function which does not depend on the variable \( x_1 \).

A product-term of an \( n \)-variable Boolean function is an expression of type \( x_1 \cdot x_2 \cdot \ldots \cdot x_k \)
\( x_k \) where \( x_k \) is either a variable \( x_i \) or its complement \( \pi x_i \), for \( 1 \leq k \leq n \) [2]. A minterm of an \( n \)-variable Boolean function is a product-term with \( k = n \). A product-term represents a subspace of an \( n \)-dimensional Boolean space, while a minterm represents a point of an \( n \)-dimensional Boolean space.

3 Composition of NLFSRs

Consider an \( n \times k \)-stage register composed of \( k \) \( n \)-stage NLFSRs \( N_1, N_2, \ldots, N_k \) as shown in Figure 1. The combining function \( f \) is a Boolean function of type \( \{0, 1\}^k \to \{0, 1\} \). A lot of research has been done on characterizing Boolean functions which are cryptographically strong [5].

\(^1\) Note that the period of a shift register is traditionally defined as the length of the longest cyclic output sequence it produces [18]. For shift registers, both definitions are equivalent. However, for general registers, in which each stage can be updated by its own function, the length of the longest state cycle can be a multiple of the length of the longest cyclic output sequence. For example, the 2-stage register with the state cycle \( ((00), (11), (01), (10)) \) of length 4 generates the cyclic output sequence \( (0, 1) \) of length 2.
It is known [9] that the composition of two cycles \( A = (S_{A,0}, S_{A,1}, \ldots, S_{A,|A|-1}) \) of length \(|A|\) and \( B = (S_{B,0}, S_{B,1}, \ldots, S_{B,|B|-1}) \) of length \(|B|\) whose states depend on disjoint sets of variables result in a set of cycles

\[
A \circ B = \bigcup_{i=0}^{d-1} D_i
\]

where \( d \) is the greatest common divisor of \(|A|\) and \(|B|\), each cycle \( D_i \) is of length \( m \), where \( m \) is the least common multiple of \(|A|\) and \(|B|\), and the \( j \)th state of \( D_i \) is a concatenation of \((j \mod |A|)\)th state of \( A \) and \(((i + j) \mod |B|)\)th state of \( B \):

\[
S_{D_i,j} = (S_{A, (j \mod |A|)} S_{B, (j + p) \mod |B|})
\]

for \( i \in \{0, 1, \ldots, d - 1\} \), \( j \in \{0, 1, \ldots, m - 1\} \).

For example, if we compose several NLFSRs whose periods are pairwise co-prime, then the resulting register has the period equal to the product of periods of the NLFSRs. Such a technique has been used to construct registers with a guaranteed long period for stream ciphers Achterbahn [15], VEST [17], and the cipher [16].

In this paper, we are focusing on the case when the NLFSRs \( N_1, N_2, \ldots, N_k \) have the same size and period which, to our best knowledge, has not been considered.

**Definition 1** An \( n \times k \)-composed register is constructed from \( k n \)-stage NLFSRs \( N_i \), \( i \in \{1, 2, \ldots, k\}, k > 1 \), such that each \( N_i \) has two cycles of the following type:

1. a cycle of length \( 2^n - 1 \) which consists of all non-zero states of \( N_i \), called *non-zero cycle*, and
2. a cycle of length one which consists of the all-zero state of \( N_i \), called *zero cycle*.

The output of an \( n \times k \)-composed register is computed from the outputs of NLFSRs \( N_1, N_2, \ldots, N_k \) using a Boolean function \( f : \{0, 1\}^k \to \{0, 1\} \).

Note that, in a traditional NLFSR, each current and next state overlap in all but one positions. The \( n \times k \)-composed registers do not match this classical definition. Therefore, their full cycles differ from the full cycles generated by NLFSRs and their output sequences are not of de Bruijn type.

In the next section, we derive a number of important properties of \((n,k)\)-composed registers.
4 Properties of \((n,k)\)-Composed Registers

Let \(x_{ij} \in \{0,1\}\) denote the state variable of the stage \(j\) of \(n\)-stage NLFSR \(N_i\), for \(i \in \{1,2,\ldots,k\}, j \in \{1,2,\ldots,n\}\). Then a state of an \(n \ast k\)-composed register \(N\) is an \(n \ast k\)-tuple of type

\[
S = (s_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, s_{2,2}, \ldots, s_{2,n}, \ldots, s_{k,1}, s_{k,2}, \ldots, s_{k,n}).
\]

The right complement of a state \(S\) of an \((n,k)\)-composed register is defined by

\[
S_R = (s_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, s_{2,2}, \ldots, s_{2,n}, \ldots, s_{k,1}, s_{k,2}, \ldots, s_{k,n}).
\]

and the left complement of a state \(S\) of an \((n,k)\)-composed register is defined by

\[
S_L = (\overline{s}_{1,1}, s_{1,2}, \ldots, s_{1,n}, s_{2,1}, s_{2,2}, \ldots, s_{2,n}, \ldots, s_{k,1}, s_{k,2}, \ldots, s_{k,n}).
\]

For the case of \(k = 1\), the right complement of \(S\) is equivalent to the companion of \(S\) and the left complement of \(S\) is equivalent to the conjugate of \(S\) [14].

Let \(S^+\) denote the next state of \(S\). Then, the following property holds.

**Lemma 1** For every state \(S\) of an \((n,k)\)-composed register, \((S_L)^+ = (S^+)_R\).

**Proof** For clarity, let us introduce an abbreviation \(s_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n})\). Then, \(S\) is of type:

\[
S = (s_{1,1}, s_1, s_{2,1}, s_2, \ldots, s_{k,1}, s_k)
\]

and the next state of \(S\) is of type:

\[
S^+ = (s_1, a_1, s_2, a_2, \ldots, s_k, a_k),
\]

for some constants \(a_1, a_2, \ldots, a_k \in \{0,1\}\).

On the other hand, the left complement of \(S\) is of type:

\[
S_L = (\overline{s}_{1,1}, s_1, s_{2,1}, s_2, \ldots, s_{k,1}, s_k)
\]

and the next state of \(S_L\) is of type:

\[
(S_L)^+ = (s_1, b_1, s_2, b_2, \ldots, s_k, b_k),
\]

for some constants \(b_1, b_2, \ldots, b_k \in \{0,1\}\).

Since all NLFSRs \(N_i\) are branchless, for all \(i \in \{1,2,\ldots,k\}\), their feedback functions \(f_i\) have the form

\[
f_i(x_{1,1}, x_{2,1}, \ldots, x_{n,1}) = x_{1,1} \oplus g_i(x_{2,1}, \ldots, x_{n,1}).
\]

Therefore, \(f_i(s_{1,1}, s_{2,1}, \ldots, s_{n,1}) \neq f_i(\overline{s}_{1,1}, s_{2,1}, \ldots, s_{n,1})\), for all \(i \in \{1,2,\ldots,k\}\). So, we can conclude that \(b_i = \overline{s}_{i,1}\) for all \(i \in \{1,2,\ldots,k\}\). This implies that \((S_L)^+ = (S^+)_R\).

The decimal representation of a state \(S = (s_{1,1}, s_{1,2}, \ldots, s_p)\) is defined as

\[
D(S) = \sum_{i=1}^{p} 2^{i-1} \ast s_i.
\]

A state of a cycle with the minimal decimal representation is called the minimal state of the cycle, denoted by \(S_{\text{min}}\).

For the case of \(k = 1\), minimal state of the cycle is equivalent to the state representative of a cycle [20].

Next, we show that, in an \((n,k)\)-composed register, \(S_{\text{min}}\) and \((S_{\text{min}})_L\) always belong to different cycles.
Fig. 2 Possible outcomes of exchanging $S^+$ and $(S_L)^+$: (a) Two cycles join into one; (b) One cycle split into two.

**Theorem 1** For every cycle of an $(n,k)$-composed register, $S_{\text{min}}$ and $(S_{\text{min}})_L$ belong to different cycles.

**Proof** Consider $(n,k+1)$-composed register $N'$ constructed from the NLFSRs $N_1, N_2, \ldots, N_k, N_{k+1}$. Since $N_1, N_2, \ldots, N_k$ induce an $(n,k)$-composed register $N$, we can partition the cycles of $N'$ into two groups:

1. Cycles obtained by composing the zero cycle of $N_{k+1}$ with all cycles of $N$.
2. Cycles obtained by composing the non-zero cycle of $N_{k+1}$ with all cycles of $N$.

For the first group, the states of all cycles of $N'$ are a concatenation of a state $S$ of a cycle of $N$ and the all-zero state of $N_{k+1}$. Therefore, for each cycle in this group, its minimal state $S'_{\text{min}}$ is a concatenation of the minimal state $S_{\text{min}}$ of the corresponding cycle of $N$ and the all-zero $n$-tuple: $S'_{\text{min}} = (S_{\text{min}}, 0, 0, \ldots, 0)$.

The left complement of $S'_{\text{min}}$ is of type:

$$(S'_{\text{min}})_L = ((S_{\text{min}})_L, 1, 0, \ldots, 0).$$

Thus $(S'_{\text{min}})_L$ belongs to the second group of cycles of $N'$.

For the second group, the minimal state $S'_{\text{min}}$ of every cycle of $N'$ is a concatenation of some state $S$ of the corresponding cycle of $N$ and the state of $N_{k+1}$ with the decimal representation 1:

$S'_{\text{min}} = (S, 1, 0, \ldots, 0)$.

Therefore, the left complement of $S'_{\text{min}}$ is a concatenation of $S_L$ and the all-zero $n$-tuple:

$$(S'_{\text{min}})_L = (S_L, 0, 0, \ldots, 0).$$

Thus $(S'_{\text{min}})_L$ belongs to the first group of cycles of $N'$.

If $S$ and its left complement $S_L$ belong to different cycles then, as shown in Figure 2a, by exchanging $S^+$ and $(S_L)^+$, we join these two cycles into one. Contrary, if $S$ and $S_L$ belong to the same cycle then, as shown in Figure 2b, by exchanging $S^+$ and $(S_L)^+$, we split this cycle into two.

**Theorem 2** An $(n,k)$-composed register has

$$C_k = \sum_{i=0}^{k-1} 2^{i+n} + 1$$

cycles, in which one cycle is a zero cycle and all other cycles are of length $2^n - 1$. 
Proof By induction of \( k \).

**Basic case:** Obviously, for the \( k = 1 \), \( C_1 = 2 \). We have one zero cycle and one non-zero cycle consisting of \( 2^k - 1 \) non-zero states.

**Induction step:** Suppose that the theorem holds for \( k \). Consider \((n, k+1)\)-composed register \( N' \) constructed from \( k+1 \) \( n \)-stage NLFSRs \( N_1, N_2, \ldots, N_k, N_{k+1} \). The NLFSRs \( N_1, N_2, \ldots, N_k \) induce an \((n, k)\)-composed register \( N \). By inductive hypothesis, \( N \) has one zero cycle and \( C_k - 1 \) other cycles of length \( 2^k - 1 \). In the proof, we refer to the latter cycles as non-zero cycles on \( N \).

We can partition the cycles of \( N' \) into four types:

1. A cycle obtained by composing the zero cycle of \( N \) with the zero cycle of \( N_{k+1} \). We get a single cycle of length one.
2. A cycle obtained by composing the zero cycle of \( N \) with the non-zero cycle of \( N_{k+1} \). We get a single cycle of length \( 2^k - 1 \).
3. Cycles obtained by composing the \( C_k - 1 \) non-zero cycles of \( N \) with the zero cycle of \( N_{k+1} \). We get \( C_k - 1 \) cycles of length \( 2^k - 1 \) each.
4. Cycles obtained by composing the \( C_k - 1 \) non-zero cycles of \( N \) with the non-zero cycle of \( N_{k+1} \). We get \((C_k - 1) + (2^n - 1)\) cycles of length \( 2^n - 1 \) each.

Thus, the total number of non-zero cycles in \( N' \) is
\[
1 + (C_k - 1) * 2^n = \sum_{i=0}^{k} 2^i = C_{k+1} - 1.
\]

As we can see from Theorem 2, the period of an \((n, k)\)-composed register is \( 2^n - 1 \). In the next section, we show how to increase this period by cycle joining.

5 Constructing Registers with Period \( O(2^{2n}) \)

In this section, we show how to obtain cycles with period \( O(2^{2n}) \) by adding to an \((n, k)\)-composed register using a logic of size \( O(n \cdot k) \).

5.1 Cycle joining transformations

First, we analyze cycles obtained by exchanging the states \( S_{\text{min}} \) and \((S_{\text{min}})_r \) of all cycles of an \((n, k)\)-composed register.

**Theorem 3** If, for every cycle of an \((n, k)\)-composed register, the successors of \( S_{\text{min}} \) and \((S_{\text{min}})_r \) are exchanged, then the resulting register has \( C_{k+1} - 1 \) cycles of length \( (2^n - 1) * 2^n \) and one cycle of length \( 2^n \).

**Proof** Consider \((n, k+1)\)-composed register \( N' \) constructed from the NLFSRs \( N_1, N_2, \ldots, N_k, N_{k+1} \). NLFSRs \( N_1, N_2, \ldots, N_k \) induce an \((n, k)\)-composed register \( N \) which, by Theorem 2, has one zero cycle and \( C_k - 1 \) other cycles of length \( 2^k - 1 \).

We partition the cycles of \( N' \) into the same four types of cycles as in the proof of Theorem 2:

1. For the cycle of type 1, its minimal state is all-zero. The left complement of the minimal state consists of \( k \) concatenated \( n \)-tuples \((100 \ldots 0)\).
2. For the cycle of type 2, its minimal state consists of \( k - 1 \) concatenated all-zero \( n \)-tuples followed by the \( n \)-tuple \((100 \ldots 0)\). The left complement of the minimal state consists of \( k - 1 \) concatenated \( n \)-tuples \((100 \ldots 0)\) followed by the all-zero \( n \)-tuple.
3. For each of the cycles of type 3, its minimal state is a concatenation of the minimal state of the corresponding cycle in \( N \) with the all-zero \( n \)-tuple. The left complement of the minimal state is a concatenation of the of the left complement of the minimal state of the corresponding cycle in \( N \) and the \( n \)-tuple \((100 \ldots 0)\).

4. For each of the cycles of type 4, its minimal state is a concatenation of some state of the corresponding cycle in \( N \) with the \( n \)-tuple \((100 \ldots 0)\). The left complement of the minimal state is a concatenation of the of the left complement of some state of the corresponding cycle in \( N \) and the all-zero \( n \)-tuple.

From the description above we can see that, for each cycle of type 4, the left complement of the minimal state of the cycle is either of type 3 or of type 1. It is of type 1 for the cycle whose minimal state consists of \( k \) concatenated \( n \)-tuples \((100 \ldots 0)\). Since there are \((C_k - 1) \ast (2^n - 1)\) cycles of type 4, we can conclude that \((C_k - 1) \ast (2^n - 1) - 1\) states in the cycles of type 3 are left complement of some minimal state of some cycle of type 4. The remaining one state in the cycles of type 3 consists of \( k - 1 \) concatenated \( n \)-tuples \((100 \ldots 0)\) followed by the all-zero state. It is the left complement of the minimal state of the cycle of type 2.

It follows from the above that if, for every cycle of \( N' \), by exchanging the next states of its \( S_{\text{min}} \) and \((S_{\text{min}})'\), we get \( C_k - 1 \) cycles of length \((2^n - 1) \ast 2^n\) which have the “flower” structure shown in Figure 3. The middle part of each “flower” corresponds to a cycle of \( N' \) of type 3. The “petal” parts are cycles of \( N' \) of type 4 or 2 (one case). We also get one cycle of length \( 2^n \) which joins the all-zero state of \( N' \) with the cycle of \( N' \) whose minimal state consists of \( k \) concatenated \( n \)-tuples \((100 \ldots 0)\).

Finally, we show that the cycles we described contain all states of \( N' \). On one hand, from Figure 3 we can conclude that the overall number of states contained in \( C_k - 1 \) “flower” cycles and the other cycles are \( A = (C_k - 1) \ast (2^n - 1) \ast 2^n + 2^n \). On the other hand, if we sum up the number of states in the cycles of types 1, 2, 3 and 4, we get \( 1 + (2^n - 1) + (C_k - 1) \ast (2^n - 1) \ast (2^n - 1) = A \).

The functions \( f(x_1, x_2, \ldots, x_n) \) and \( f(x_1, x_2, \ldots, x_n) \oplus \bar{x}_1 x_2 \ldots x_n \), where \( \bar{x}_1 \) is defined as:

\[
\bar{x}_i = \begin{cases} 
    \bar{x}_i, & \text{if } s_i = 0 \\
    x_i, & \text{if } s_i = 1
\end{cases}
\]

evaluate to the same values for all assignments of their variables except \((s_1, s_2, \ldots, s_n)\). This implies that we can change the next state of a state \((s_1, s_2, \ldots, s_n)\) of an \( n \)-stage NLFSR with the feedback function \( f(x_1, x_2, \ldots, x_n) \) from \((s_2, \ldots, s_n, f(s_1, x_2, \ldots, s_n))\) to \((s_2, \ldots, s_n, \bar{f}(x_1, x_2, \ldots, s_n))\) by adding to \( f \) the minterm \( x_1 x_2 \ldots x_n \). Consequently, an \((n, k)\)-composed
register in which the states $S^\text{min}_n$ and $(S^\text{min}_n)_L$ are exchanged can be obtained by adding to the feedback function of every NLFSR $N_i$ the minterms corresponding to the states $S^\text{min}_n$ and $(S^\text{min}_n)_L$.

5.2 Extra logic block

Next, we derive sum-of-product expressions representing the $S^\text{min}_n$ and $(S^\text{min}_n)_L$ of all cycles of an $(n,k)$-composed register.

**Theorem 4** The set of minimal states of all cycles of an $(n,k)$-composed register is represented by the following Boolean function:

$$f^\text{min}_n(x_1, x_2, \ldots, x_{nk}) = \bigwedge_{i=0}^{n-1} (x_{i+n+1} \cdot \prod_{j=i+2}^{nk} x_j) \lor \prod_{j=1}^{nk} x_j$$

(1)

and the set of left complements of its minimal states is represented by:

$$f^\text{min}_L(x_1, x_2, \ldots, x_{nk}) = \bigwedge_{m=0}^{n-1} (\prod_{j=m+1}^{nk} x_j \cdot \prod_{p=0}^{k-1} (x_{n+p+1} \cdot \prod_{m=n+p+2}^{nk} x_m)) \lor \prod_{p=0}^{k-1} (x_{n+p+1} \cdot \prod_{m=n+p+2}^{nk} x_m)$$

(2)

where "\lor" stands for the Boolean OR.

**Proof** By induction of $k$.

**Basic case:** $k = 1$, i.e. $N = N_1$. Then the equation (1) reduces to

$$f^\text{min}_n(x_1, x_2, \ldots, x_n) = (x_1 \cdot \prod_{j=2}^{n} x_j) \lor \prod_{j=1}^{n} x_j$$

which correspond to the minimal state $(1, 0, \ldots, 0)$ of the cycle of length $2^n - 1$ of $N_1$ and to the minimal state $(0, 0, \ldots, 0)$ of the zero cycle of $N_1$.

The equation (2) reduces to

$$f^\text{min}_L(x_1, x_2, \ldots, x_n) = \prod_{j=1}^{n} x_j \lor (x_1 \cdot \prod_{m=2}^{n} x_m)$$

which correspond to the left complements of the minimal states of the cycle of length $2^n - 1$ of $N_1$ and zero cycle of $N_1$, respectively.

**Induction step:** Suppose that the theorem holds for $k$. Consider $(n,k+1)$-composed register $N'$ constructed from $k+1$ $n$-stage NLFSRs $N_1, N_2, \ldots, N_k, N_{k+1}$. The NLFSRs $N_1, N_2, \ldots, N_k$ induce an $(n,k)$-composed register $N$. By inductive hypothesis, their set of minimal states is represented by the equation (1). The product-terms of the equation (1) are listed in the following table with $k+1$ rows:

<table>
<thead>
<tr>
<th>$S^\text{min}_n$</th>
<th>Corresponding product-term</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 \ldots 0</td>
<td>$x_1 x_2 \ldots x_{nk}$</td>
</tr>
<tr>
<td>1 0 0 \ldots 0</td>
<td>$x_1 x_2 \ldots x_{nk}$</td>
</tr>
<tr>
<td>x 1 0 \ldots 0</td>
<td>$x_{n+1} x_{n+2} \ldots x_{nk}$</td>
</tr>
<tr>
<td>x x 1 \ldots 0</td>
<td>$x_{2n+1} x_{2n+2} \ldots x_{nk}$</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>x x x \ldots 1</td>
<td>$x_{(k-1)n+1} x_{(k-1)n+2} \ldots x_{nk}$</td>
</tr>
</tbody>
</table>
Cycles obtained by composing the non-zero cycle of \( N \) all possible combinations which can be obtained from the cycles of \( N \) into two groups:

- \( S_{\text{min}} \) of the minimal states of all cycles in the first group is composed of all possible states of \( N \) and represent the set of minimal states of all cycles in the second group is composed of all possible states of \( N \), i.e. it is of type \( S_{\text{min}} = (S_{\text{min}} 0) \). Therefore, the Boolean function \( f_{\text{min}}^1 \) representing the set of minimal states of all cycles in the first group is of type

\[
f_{\text{min}}^1 = \bigvee_{i=1}^{k+1} p_i \cdot x_{km+1} x_{km+2} \cdots x_{kn(k+1)}
\]

where \( p_i \) is the product-term from the \( i \)th row of the table above.

For each cycle in the second group, its minimal state \( S_{\text{min}}^\prime \) is a concatenation of some state \( S \) of the corresponding cycle of \( N \) and the state of \( N_{k+1} \) with the decimal representation 1, i.e. it is of type \( S_{\text{min}}^\prime = (S_{\text{min}} 1) \). Since cycles an \( (n+1,k) \)-composed register \( N' \) represent all possible combinations which can be obtained from the cycles of \( N_1, N_2, \ldots, N_k \), the set of minimal states in the second group is composed of all possible states of \( (n,k) \)-composed register \( N \) and 1. Therefore, the Boolean function \( f_{\text{min}}^2 \) representing the set of minimal states of all cycles in the second group is of type

\[
f_{\text{min}}^2 = x_{km+1} x_{km+2} \cdots x_{kn(k+1)}
\]

By taking the Boolean OR of \( f_{\text{min}}^1 \) and \( f_{\text{min}}^2 \), we get equation (1) for \( k = k + 1 \).

Since the left complement of an \( n \)-tuple \( 0 \) is 1 and vice-versa, for the left complements of the minimal states of \( (n,k) \)-composed register \( N \) we get the product-terms listed in Table 1.

For the cycles in the first group, the set of left complements of the minimal states \( (S_{\text{min}})_{L} = ((S_{\text{min}})_L, 1) \), for all left complements of the minimal states \( (S_{\text{min}})_L \) of \( N \). Therefore, the Boolean function \( f_{\text{mid}}^1 \) representing the set of left complements of the minimal states of all cycles in the first group is of type

\[
f_{\text{mid}}^1 = \bigvee_{i=1}^{k+1} p_i \cdot x_{km+1} x_{km+2} \cdots x_{kn(k+1)}
\]

where \( p_i \) is the product-term from the \( i \)th row of the table above.

---

**Table 1** Product-terms representing left complements of the minimal states.

<table>
<thead>
<tr>
<th>((S_{\text{min}})_L)</th>
<th>Corresponding product-term</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 ... 1</td>
<td>( x_1 x_2 \cdots x_{km+1} x_{km+2} \cdots x_{kn(k+1)} )</td>
</tr>
<tr>
<td>0 1 1 ... 1</td>
<td>( x_1 x_2 \cdots x_{km+1} x_{km+2} \cdots x_{kn(k+1)} )</td>
</tr>
<tr>
<td>x 0 1 ... 1</td>
<td>( x_1 x_2 \cdots x_{km+1} x_{km+2} x_{km+3} x_{km+4} x_{km+5} x_{km+6} x_{km+7} x_{km+8} x_{km+9} )</td>
</tr>
<tr>
<td>x x 0 ... 1</td>
<td>( x_1 x_2 \cdots x_{km+1} x_{km+2} x_{km+3} x_{km+4} x_{km+5} x_{km+6} x_{km+7} x_{km+8} x_{km+9} )</td>
</tr>
<tr>
<td>...</td>
<td>( x_1 x_2 \cdots x_{km+1} x_{km+2} x_{km+3} x_{km+4} x_{km+5} x_{km+6} x_{km+7} x_{km+8} x_{km+9} )</td>
</tr>
<tr>
<td>x x x ... 0</td>
<td>( x_{(k-1)km+1} x_{(k-1)km+2} \cdots x_{km(k+1)} )</td>
</tr>
</tbody>
</table>

In this table, \( 0 \) denotes an \( n \)-tuple consisting of all zeros: \( 0 = (0,0,\ldots,0) \), \( 1 \) denotes an \( n \)-tuple consisting of 1 followed by \( n-1 \) zeros: \( 1 = (1,0,\ldots,0) \), and \( x \) denotes an \( n \)-tuple consisting of all \( x \) such that \( x = (x,x,\ldots,x) \), where the symbol "x" means either 0 or 1.

As in the proof of Theorem 1, we partition the cycles of \((n,k+1)\)-composed register \( N'\) into two groups:

1. Cycles obtained by composing the zero cycle of \( N_{k+1} \) with all cycles of \( N \).
2. Cycles obtained by composing the non-zero cycle of \( N_{k+1} \) with all cycles of \( N \).
For the cycles in the second group, the set of left complements consists of states \((S'_{\text{min}})_L = (S_L, 0)\), where \(S_L\) a left complement of a state \(S\) of \(N\), for all possible states \(S\) of \(N\). Therefore, the Boolean function \(f'_{\text{min}}\) representing the set of left complements of the minimal states of all cycles in the second group is of type

\[
f'_{\text{min}} = \overline{x_{kn+1}}x_{kn+2} \cdots x_{kn+k+1}.
\]

By taking the Boolean OR of \(f'_{\text{min},1}\) and \(f'_{\text{min},2}\), we get equation (2) for \(k = k + 1\).

Some optimizations can be done to reduce the size of the expression \(f_{\text{min}} \lor f'_{\text{min}}\). First, the product-terms in the first two rows of both tables in the proof of Theorem 3 can be joined into product-terms \(x_2 \cdots x_{n+k}\) and \(x_2 \cdots x_{n+k+1} x_3 \cdots x_{n+k+2} \cdots x_{n+k}\). In the proof, we kept them separately in order to give a better view on the structure of \(f_{\text{min}}\) and \(f'_{\text{min}}\). Also, the product-terms in the last rows of both tables can be joined into the product-term \(x_{2n+1}^1 \cdots x_{n+k}\). In this way, the overall number of product-terms in \(f_{\text{min}} \lor f'_{\text{min}}\) can be reduced to \(2^k + 1\).

Furthermore, note that the product-terms of \(f_{\text{min}} \lor f'_{\text{min}}\) have common sub-terms \(x_2 \cdots x_n, x_{n+2} \cdots x_{2n}, \cdots, x_{(k-1)n+2} \cdots x_{nk}\). Each of these common sub-terms can be represented only once and shared by all product-terms. In this way, the number of literals \(x_i\) in the representation of \(f_{\text{min}} \lor f'_{\text{min}}\) can be reduced to \(O(n^k)\). So, the block \(f_{\text{min}} \lor f'_{\text{min}}\) can be implemented with \(O(n^k)\) 2-input gates.

The general structure of a register with period \((2^n - 1) \cdot 2^k\) constructed using Theorem 3 is shown in Figure 4.

6 Constructing Registers with the Maximum Period

In this section, we show how to join cycles of a register constructed using Theorem 3 into one cycle.

6.1 Cycle joining transformations

The approach described in the previous section allows us to construct \(n \cdot k\)-stage registers which have \(\sum_{i=0}^{i=n} 2^m\) cycles of length \((2^n - 1) \cdot 2^m\) and one cycle of length \(2^n\). The following Theorem shows how to join these cycles into one.

**Theorem 5** Let \(N\) be an \(n \cdot k\)-stage register constructed using Theorem 3. If the following transformations are applied to each current state \(S\) of \(N\) before computing the next state:

\[T1: \text{If } D(S) \in \{2^m, 2^m+1, \ldots, 2^m+2\}, \text{ for } i \in \{1, 2, \ldots, k-2\}, \text{ then } S \text{ is transformed to the state } S' \text{ such that } D(S') = D(S) + 1.\]

\[T2: \text{If } D(S) = 2^m+1 \text{ for } i \in \{0, 1, \ldots, k-3\}, \text{ then } S \text{ is transformed to the state } S' \text{ such that } D(S') = 2^{(i+1)n}.\]

\[T3: \text{If } D(S) = 2^{(k-2)n+1} - 1, \text{ then } S \text{ is transformed to the state } S' \text{ such that } D(S') = 0.\]

\[T4: \text{If } D(S) = 0, \text{ then } S \text{ is transformed to the state } S' \text{ such that } D(S') = D(S) + 1.\]

then the resulting register has period \(2^m\).

\[\text{A 2-input gate implements a binary Boolean operation of type } \{0, 1\}^2 \rightarrow \{0, 1\}.\]
Consider an $n \times k$-stage register $N$ constructed using Theorem 3. $N$ has $\sum_{i=0}^{k-2} 2^{i+1}$ cycles of length $(2^n - 1) \times 2^n$ and one cycle of length $2^n$. The cycles of length $(2^n - 1) \times 2^n$ have the "flower" structure shown in Figure 3.

We can connect all cycles of $N$ together by transforming the minimal state of the middle cycle of each "flower" into the minimal state of the middle cycle of the next "flower" before computing the next state of $N$, and finally appending to the resulting chain of "flowers" the cycle of length $2^n$, as shown in Figure 5.

From the proofs of Theorems 3 and 4, it follows that the set of minimal states of the middle cycles of "flowers" is given by

\[
\begin{array}{cccccccc}
1 & 2 & 3 & \ldots & k-1 & k \\
1 & 0 & 0 & \ldots & 0 & 0 \\
x & 1 & 0 & \ldots & 0 & 0 \\
x & x & 1 & \ldots & 0 & 0 \\
\vdots \\
x & x & x & \ldots & 1 & 0
\end{array}
\]

The table above contains $k-1$ rows. For each $i \in \{0, 1, \ldots, k-2\}$, the row $i$ represents a block of $2^{i+1}$ minimal states of the middle cycle of a "flower" with the decimal representations $\{2^{i+1}, 2^{i+1} + 1, \ldots, 2^{i+1} + 2^{i+1} - 2\}$. We order the states in each block according to their $D(S)$. Then, for $i = \{1, 2, \ldots, k-2\}$, we can visit all states in a block by incrementing by 1 $D(S)$ of each state $S$ in the block but the last. This is done by the transformation $T1$. To visit all blocks, for $i = \{0, 1, \ldots, k-3\}$, we go from the last state of the block $i$, which has $D(S) = 2^{i+1} + 1$, to the first state of the block $i+1$, which has $D(S') = 2^{(i+1)+1}$. This is done by the transformation $T2$.

In order to append the cycle of length $2^n$ to the resulting chain of "flowers", we go from the last state of the last block, which has $D(S) = 2^{(k-2)+1} + 1$, to the minimal state of the cycle of length $2^n$, which is the all-0 state. This is done by the transformation $T3$. Finally, we close the cycle by the going from the all-zero state to first state of the first block. This is done by incrementing $D(S)$ of all-0 state by 1, i.e. by the transformation $T4$.

Thus, by applying the transformation’s $T1, T2, T3$ and $T4$, we join all cycles of $N$ into one.
logic can also be implemented with $O$ in complementing values of all stages controlled by these ORs. We need $O$ terms evaluates to 1, the outputs of ORs fed by this product-term become 1. That results

$T_3$ and $O$

transformations $T_2$

ling ORs and each OR can have up to $T_1$, $T_2$, $T_3$

The adder is controlled by an OR-gate with inputs from the product-terms implementing the set states with $D$

$T_4$

so, the overall complexity of implementing the transformations $T_1$, $T_2$, $T_3$ and $T_4$ is $O(n + k^2)$ 2-input gates. Note that many of the product-terms used for implementing $T_1$, $T_2$, $T_3$ and $T_4$ are also used for implementing $f_{\text{min}}$ and $f_{\text{max}}$, so they can be shared.

As an example, consider the simple case of $k = 2$ for which we only need to transform the state $(1, 0)$ into $(0, 0)$ and vice versa. This can be done using an $n \times k - 1$ input NOR gate with inputs from all the stages but $(1, 1)$ and a 2-input XOR gate with inputs from the stage $(1, 1)$ and the NOR gate and the output to the stage $(1, 1)$. In this way, we complement the value of the stage $(1, 1)$ only if all the remaining stages have the value 0. Since $x_1 + x_3 + \ldots + x_{nk} = \overline{x_1} \overline{x_3} \ldots x_{nk}$ and the product-term $\overline{x_1} \overline{x_3} \ldots x_{nk}$ is used for implementing the function $f_{\text{min}}$, it brings no extra cost.

In the next section, we show that, in the general case, the transformations $T_1$, $T_2$, $T_3$ and $T_4$ require $O(n + k^2)$ 2-input gates to be implemented.

The registers constructed using Theorem 5 require two time steps to compute the next step from a current step. At the first step, we update the current state by applying the transformations $T_1$, $T_2$, $T_3$ and $T_4$. At the second step, we compute the next state from the resulting updated state.

6.2 Extra logic block

In the general case of $k > 2$, we can implement the transformations $T_1$ and $T_4$ using an $(k - 2) \times n$-bit adder in which one of the operands is 0 or 1, supplied by a controlling OR described below, and another operand is the content of the stages of the first $k - 2$ n-bit NLFSRs. The result of addition is fed back into the stages of the first $k - 2$ n-bit NLFSRs. The adder is controlled by an OR-gate with inputs from the product-terms implementing the set states with $D(S) \in \{2^{i\alpha n}, 2^{i\alpha n + 1}, \ldots, 2^{i\alpha n + 1 - 2}\}$, for $i = \{1, 2, \ldots, k - 2\}$, and the all-zero state. The addition of 1 is performed only if one of the product-terms evaluates to 1. A $(k - 2) \times n$-bit adder can be implemented with $O(n + k)$ 2-input gates. The controlling logic can also be implemented with $O(n + k)$ 2-input gates.

The transformations $T_2$ and $T_3$ can be implemented by assigning to each stage $(i, j)$, for every $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, k - 2\}$, and for $i = 1$ and $j = k - 1$, a 2-input XOR gate, with inputs from the stage $(i, j)$ and a controlling OR-gate described below and the output to the stage $(i, j)$. The OR-gates take inputs from the product-terms implementing the set states with $D(S) = 2^{i\alpha n + 1 - 1}$ for $i = \{0, 1, \ldots, k - 2\}$. Only if one of the productterms evaluates to 1, the outputs of ORs fed by this product-term become 1. That results in complementing values of all stages controlled by these ORs. We need $O(n + k)$ controlling ORs and each OR can have up to $k - 1$ inputs. Therefore, the controlling logic for the transformations $T_2$ and $T_3$ requires $O(n + k^2)$ 2-input gates.

So, the overall complexity of implementing the transformations $T_1$, $T_2$, $T_3$ and $T_4$ is $O(n + k^2)$ 2-input gates. Note that many of the product-terms used for implementing $T_1$, $T_2$, $T_3$ and $T_4$ are also used for implementing $f_{\text{min}}$ and $f_{\text{max}}$, so they can be shared.
7 Conclusion

In this paper, we presented a method for constructing $n \times k$-stage registers with period $2^{n}k$ by a composition of $k \ n$-stage NLFSRs. First, we show that an $n \times k$-stage register with period $O(2^{2n})$ can be constructed from $k \ n$-stage NLFSRs by adding to their feedback functions a logic block of size $O(n \times k)$. Second, we show how to join all cycles into one by using one more logic block of size $O(n \times k^2)$ and an extra time step.

Future work involves security analysis of the presented method.

Acknowledgements This work was supported in part by the project No 621-2010-4388 from Swedish Research Council.

References

On Discrete Fourier Transform, Ambiguity, and Hamming-Autocorrelation of Pseudorandom Sequences

Gottlieb Pirsic · Arne Winterhof

Abstract We estimate discrete Fourier transform, ambiguity, and Hamming-autocorrelation of m-ary sequences in terms of their (periodic) correlation measure of order 4. Roughly speaking, we show that every pseudorandom sequence, that is, any sequence with small correlation measure up to a sufficiently large order, cannot have a large discrete Fourier transform, ambiguity, or Hamming-autocorrelation. Conversely, there are sequences, for example the two-prime generator, with large correlation measure of order 4 but small discrete Fourier transform, ambiguity, autocorrelation, and Hamming-autocorrelation.

Keywords Pseudorandom sequences · Correlation · Autocorrelation · Discrete Fourier Transform · Ambiguity

1 Introduction

Mauduit and Sárközy introduced a measure of pseudorandomness for finite sequences (first for binary sequences in [8] and later for m-ary sequences in [9]). We define an analog for periodic sequences.

Let $\mathcal{E}_m = \{ \varepsilon_1, \ldots, \varepsilon_m \}$ be the set of the complex m-th roots of unity and $\mathcal{F}$ be the set of m! bijections of $\mathcal{E}_m$. Consider a $T$-periodic sequence $E_T = (e_0, e_1, \ldots)$ over $\mathcal{E}_m$, that is, $(e_0, e_1, \ldots, e_{T-1}) \in \mathcal{E}_m^T$ and $e_{n+T} = e_n$ for $n = 0, 1, \ldots$. The (periodic) correlation measure of order $\ell$ is

$$\Gamma_{\ell}(E_T) = \max_{\phi, D} \left| \sum_{n=0}^{T-1} \phi_1(e_{n+d_1}) \phi_2(e_{n+d_2}) \cdots \phi_\ell(e_{n+d_\ell}) \right|,$$

The research of G. Pirsic was supported by the Austrian Science Fund (FWF), P23285-N18

G. Pirsic
Institute for Financial Mathematics, Kepler University Linz, Altenbergerstr. 69, A-4040 Linz, Austria, E-mail: g.pirsic@gmail.com

A. Winterhof
Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Altenberger Str. 69, A-4040 Linz, Austria, E-mail: arne.winterhof@oeaw.ac.at
where the maximum is taken over all \( \phi = (\varphi_1, \ldots, \varphi_\ell) \in \mathcal{F}^\ell \) and \( D = (d_1, \ldots, d_\ell) \) with \( 0 \leq d_1 < \ldots < d_\ell < T \).

We call a sequence \( E_T \) pseudorandom if \( \Gamma_\ell(E_T) \) is small up to a sufficiently large \( \ell \).

Heuristically, the significance of the correlation measure lies in assessing the dependence of \( \ell \) concurrent distortions of some wirelessly transmitted signal stream. Through reflections and Doppler effects there may arise, for example, time shifts or phase distortions, which are modeled by \( D \) and \( \phi \) respectively. In most cases, however, it is not feasible to treat the fully general case, therefore usually specially relevant subclasses of \( \mathcal{F} \) are chosen, most importantly phase shifts and conjugation. Also the order \( \ell \) is commonly considered a small integer such as 1, 2, or 4.

We can estimate \( \Gamma_2(E_T) \) in terms of \( \Gamma_4(E_T) \) (cf. [2] for the case of finite binary sequences):

\[
\Gamma_2(E_T) \ll T^{1/2} \Gamma_4(E_T)^{1/2},
\]

(1)

Here \( A \ll B \) means \( A \leq cB \) with an absolute constant \( c > 0 \). For the convenience of the reader we will give the short proof of (1) in Section 2.

Other figures of merit of a sequence can be estimated in terms of the correlation measure of order \( k \) as for example its linear complexity [1,3] or, trivially, its maximum autocorrelation

\[
C(E_T) = \max_{1 \leq t < T} \left| \sum_{n=0}^{T-1} e_n e_{n+t} \right| \leq \Gamma_2(E_T) \ll T^{1/2} \Gamma_4(E_T)^{1/2},
\]

(2)

where \( \overline{x} \) is the complex conjugate of \( x \), that is, \( C(E_T) \) corresponds to the special case of \( \ell = 2 \) and \( \varphi_1, \varphi_2 \) being the identity and complex conjugation, respectively.

In this paper we prove upper bounds on some other measures of pseudorandomness of a \( T \)-periodic \( m \)-ary sequence (over \( \mathcal{E}_m \)) in terms of its autocorrelation and correlation measures of order 2 and 4, respectively:

- discrete Fourier transform,
- ambiguity,
- Hamming-autocorrelation.

Sequences with good correlation properties find many applications in wireless communications. In particular, small discrete Fourier transforms are needed in orthogonal frequency division multiplexing (OFDM) systems and low ambiguities for radar systems and signal processing schemes, see for example [5]. A small Hamming-autocorrelation is important for frequency hopping sequences, see for example [7].

The maximum discrete Fourier transform of \( E_T \) is

\[
D(E_T) = \max_{0 \leq k < T} \left| \sum_{n=0}^{T-1} e_n \omega_T^{-kn} \right|,
\]

where \( \omega_T \) is any primitive complex \( T \)th root of unity. In Section 3 we prove

\[
D(E_T) \ll T^{1/2} C(E_T)^{1/2} \ll T^{1/2} \Gamma_2(E_T)^{1/2} \ll T^{3/4} \Gamma_4(E_T)^{1/4},
\]

(3)

where the second and third inequality follow from (2).
The maximum ambiguity of $E_T$ is defined as

$$A(E_T) = \max_{1 \leq t, k < T} \left| \sum_{n=0}^{T-1} e_{n+t} e_{n+k} \omega_T^{-kn} \right|.$$  

In Section 4 we prove

$$A(E_T) \ll T^{1/2} \Gamma_4(E_T)^{1/2}. \quad (4)$$

The maximum Hamming-autocorrelation of $E_T$ is

$$H(E_T) = \max_{1 \leq t < T} \sum_{n=0}^{T-1} h(e_n, e_{n+t}),$$

where $h(x, y) = 1$ if $x = y$ and $h(x, y) = 0$ otherwise. For prime $m$ we prove

$$\left\lceil \frac{T}{m} \right\rceil - 1 \leq H(E_T) \leq \frac{T}{m} + \frac{m-1}{m} \Gamma_4(E_T) \ll \frac{T}{m} + T^{1/2} \Gamma_4(E_T)^{1/2}, \quad m \in \mathbb{P}, \quad (5)$$

in Section 5.

Consequently, any pseudorandom sequence $E_T$, in particular any sequence $E_T$ with small $\Gamma_4(E_T)$, cannot have a large autocorrelation, maximum discrete Fourier transformation, ambiguity, or Hamming-correlation.

Conversely, even if a sequence has small maximum autocorrelation, discrete Fourier transformation, ambiguity, and Hamming-correlation, it can have a large correlation measure of order 4 and be not pseudorandom. An example of such a sequence is the two-prime generator (or Jacobi sequence), that is, a quadratic character sequence modulo $pq$ with two different primes $p$ and $q$. Moreover, there are sequences, for example character sequences modulo $p$, for which direct estimates are better than the results which can be obtained with (2)-(5) from bounds on the correlation measure of order 4. However, if we substitute $\omega_T$ by any other primitive $H$th root of unity $\omega_H$ with $H \neq T$ in the definitions of $D(E_T)$ and $A(E_T)$ the direct methods fail whereas (3) and (4) are still valid and can give nontrivial estimates. Actually, the discrete Fourier transform and ambiguity are very often defined in this more general way, see [6].

We present the results on character sequences modulo $p$ and modulo $pq$ with primes $p \neq q$ in Section 6.

## 2 Correlation measures of order 2 and 4

In this section we prove (1).

For any positive integer $J$ we have

$$J \left| \sum_{n=0}^{T-1} \varphi_1(e_{n+d_1}) \varphi_2(e_{n+d_2}) \right| \leq \sum_{j=0}^{J-1} \sum_{n=0}^{T-1} \varphi_1(e_{n+j+d_1}) \varphi_2(e_{n+j+d_2}) \leq W,$$

where

$$W = \left| \sum_{n=0}^{T-1} \sum_{j=0}^{J-1} \varphi_1(e_{n+j+d_1}) \varphi_2(e_{n+j+d_2}) \right|.$$
The Cauchy-Schwarz inequality implies
\[
W^2 \leq T \sum_{n=0}^{T-1} \sum_{j=0}^{J-1} \varphi_1(e_{n+j+d_1}) \varphi_2(e_{n+j+d_2})^2
= T \sum_{j=0}^{J-1} \sum_{n=0}^{T-1} \varphi_1(e_{n+j+d_1}) \varphi_2(e_{n+j+d_2}) \overline{\varphi_1(e_{n+l+d_1})} \overline{\varphi_2(e_{n+l+d_2})},
\]
where \( \overline{\varphi(x)} = \overline{\varphi(x)} \).

For \( l = j, l = j + d_2 - d_1 \), and \( l = j + d_1 - d_2 \) we estimate the absolute value of the inner sum trivially by \( T \). For the remaining pairs \((i,j)\) we can bound the inner sum by \( \Gamma_4(E_T) \) and get
\[
W^2 \leq T(3JT + J^2 \Gamma_4(E_T)).
\]
Choosing
\[
J = \left\lfloor \frac{T}{\Gamma_4(E_T)} \right\rfloor
\]
we get
\[
\left| \sum_{n=0}^{T-1} \varphi_1(e_{n+d_1}) \varphi_2(e_{n+d_2}) \right| \leq 2T^{1/2} \Gamma_4(E_T)^{1/2}.
\]

3 Discrete Fourier transformation and autocorrelation

In this section we prove (3).

For \( 0 \leq k < T \), put
\[
W_k = \left| \sum_{n=0}^{T-1} e_n \omega^{-kn} \right|,
\]
such that
\[
D(E_T) = \max_{0 \leq k < T} W_k.
\]

For any positive integer \( J \) we have
\[
JW_k = \left| \sum_{j=0}^{J-1} \sum_{n=0}^{T-1} e_{n+j} \omega^{-j(n+j)} \right|
\leq \sum_{n=0}^{T-1} \sum_{j=0}^{J-1} e_{n+j} \omega^{-jk}.
\]

Applying the Cauchy-Schwarz inequality we get
\[
J^2 W_k^2 \leq T \sum_{n=0}^{T-1} \sum_{j=0}^{J-1} |e_{n+j} \omega^{-jk}|^2
\]
On DFT, Ambiguity and Hamming-Autocorrelation of Pseudorandom Sequences

\[ T^{J-1} \sum_{j,l=0}^{T-1} e_{n+j} e_{n+l} \omega_T^{(l-j)k} \]

\[ \leq T \left( JT + J^2 \max_{j \neq l} \left| \sum_{n=0}^{T-1} e_{n+j} e_{n+l} \right| \right) \]

\[ \leq T \left( JT + J^2 C(E_T) \right). \]

Choosing \( J = \lceil T/C(E_T) \rceil \) we get the result.

4 Ambiguity and correlation measure of order 4

In this section we prove (4).

For \( 1 \leq t, k \leq T \) put

\[ W_{t,k} = \left| \sum_{n=0}^{T-1} e_n e_n + t \omega_T^{-kn} \right|. \]

For any positive integer \( J \) we have

\[ JW_{t,k} = \left| \sum_{n=0}^{T-1} e_{n+j} e_{n+j+t} \omega_T^{-(n+j)k} \right| \]

\[ \leq \left| \sum_{n=0}^{T-1} \sum_{j=0}^{J-1} e_{n+j} e_{n+j+t} \omega_T^{-jk} \right|. \]

Applying the Cauchy-Schwarz inequality we get

\[ J^2 W_{t,k}^2 \leq T \sum_{n=0}^{T-1} \left| \sum_{j=0}^{J-1} e_{n+j} e_{n+j+t} \omega_T^{-jk} \right|^2 \]

\[ = T \sum_{j,l=0}^{T-1} e_{n+j} e_{n+l} \omega_T^{(l-j)k} \]

\[ \leq T \left( 3JT + J^2 \max_{j \notin \{t,l,l+t\}} \left| \sum_{n=0}^{T-1} e_{n+j} e_{n+l} e_{n+l+t} \right| \right) \]

\[ \leq T \left( 3JT + J^2 \Gamma_4(E_T) \right). \]

Choosing \( J = \lceil T/\Gamma_4(E_T) \rceil \) we get the result.
5 Hamming-autocorrelation

In this section we prove (5).

For \( x, y \in \mathcal{E}_m \) we have
\[
h(x, y) = \frac{1}{m} \sum_{j=0}^{m-1} (xy^{-1})^j
\]
and thus since \( \pi = x^{-1} \), for any \( 1 \leq t < T \),
\[
\sum_{n=0}^{T-1} h(e_n, e_{n+t}) = \frac{1}{m} \sum_{j=0}^{m-1} \sum_{n=0}^{T-1} e_n e_{n+t}^j.
\]
Hence,
\[
\left| H(E_T) - \frac{T}{m} \right| \leq \frac{m-1}{m} \max_{1 \leq j < m} C(E^j_T), \quad (6)
\]
where \( E^j_T = (e_0^j, e_1^j, \ldots) \).

If \( m \) is prime, each mapping \( x \mapsto x^j \) with \( 1 \leq j < T \) is a permutation of \( \mathcal{E}_m \) and the maximum on the right hand side can be bounded by \( \Gamma_2(E_T) \).

In the important case of quaternary sequences, that is \( m = 4 \), we can prove
\[
\left| H(E_T) - \frac{T}{4} \right| \leq \frac{3}{2} \Gamma_2(E_T), \quad m = 4. \quad (5')
\]

For \( j \in \{1, 3\} \) we can estimate \( C(E^j_T) \) by \( \Gamma_2(E_T) \) since \( x \mapsto x^j \) is a permutation. For \( j = 2 \) we have
\[
x^2 = \frac{1-i}{2} \tau(x) + \frac{1+i}{2} \tau(x), \quad x \in \mathcal{E}_4 = \{1, -1, i, -i\},
\]
where \( \tau \) is the transposition \( -1 \leftrightarrow i \). Hence,
\[
x^2 y^2 = \frac{1}{2} (-i \tau(x) \tau(y) + \tau(x) \tau(y) + \tau(x) \tau(y) + i \tau(x) \tau(y))
\]
and thus
\[
\left| \sum_{n=0}^{T-1} e_n^2 e_{n+t}^2 \right| \leq \left| \sum_{n=0}^{T-1} \tau(e_n) \tau(e_{n+t}) \right| + \left| \sum_{n=0}^{T-1} \tau(e_n) \tau(e_{n+t}) \right| \leq 2 \Gamma_2(E_T).
\]

Note that in [4, Corollary 1.2] the lower bound
\[
H(E_T) \geq \left\lfloor \frac{T}{m} \right\rfloor - 1
\]
is shown and our bound is close to be optimal. (See also the Lempel-Greenberger bound [7].)
6 Examples

First, as a trivial example we mention that for the constant sequence our bounds (1)-(5) are sharp (but trivial). In the following we present some typical nontrivial sequences exhibiting various behaviors in the relation between the different measures.

6.1 Character sequences modulo $p$

The following example shows that for some sequences our bounds give weaker results than direct estimates.

Let $p$ be a prime and $\chi$ a multiplicative character modulo $p$ of order $m > 1$ (that is $m | p - 1$) with the convention $\chi(0) = 1$. Consider the $p$-periodic sequence $E_T = (e_0, e_1, \ldots)$ defined by

$$e_n = \chi(n), \quad n = 0, 1, \ldots$$

For $j$ with $\gcd(j, m) = 1$ the mapping $x \mapsto x^j$ is a permutation of $E_m$ and by the Weil bound we know only

$$\left| \sum_{n=0}^{p-1} \chi(n + d_1)^{j_1} \cdots \chi(n + d_k)^{j_k} \right| \ll p^{1/2}$$

and we are not able to estimate $I_4(E_p)$ by a bound better than $\ll p^{1/2}$. Therefore (2)-(5) do not give anything better than $\ll p^{3/4}$.

Note that for $m \geq 4$ one can derive a bound on $I_4(E_p)$ of order of magnitude $p^{1/2}$ adapting the ideas of [9], where the implied constant depends on $m$. For $m \in \{2, 3\}$ all permutations of $E_m$ are of the form $x \mapsto \omega_m^i x^j$, $i = 0, 1, \ldots, m - 1$, $j = 1, 2, \ldots, m - 1$ and we get immediately $I_4(E_T) \ll p^{1/2}$.

However, direct application of the Weil bound for hybrid character sums (see [11, Appendix 5, Example 12] or [10, Lemma 3.1])

$$\left| \sum_{n=0}^{p-1} \chi(f(n))\omega_p^{g(n)} \right| \ll (\deg(f) + \deg(g))p^{1/2}$$

if either $f(X)$ is not of the form $ah(X)^s$ where $s > 1$ is the order of $\chi$ or $g(X)$ is not constant (we do not need the restriction $g(X) \neq c(h(X)^p - h(X))$ since then if $g(X)$ is not constant, we have $\deg(g) \geq p$ and the bound is trivial), and the well known relation

$$\sum_{n=0}^{p-1} \chi(n)\chi(n+a) = -1 + \chi(a) + \chi(-a), \quad \gcd(a, p) = 1,$$

provide

$$C(E_p) = \max_{1 \leq t < p} \left| \sum_{n=0}^{p-1} \chi(n)\chi(n+t) \right| \leq 3, \quad (7)$$

$$D(E_p) = \max_{0 \leq k < p} \left| \sum_{n=0}^{p-1} \chi(n)\omega_p^{-kn} \right| \ll p^{1/2},$$
\[ A(E_p) = \max_{1 \leq k, t < p} \left| \sum_{n=0}^{p-1} \chi(n)\chi(n+t)\omega^{-kn}_p \right| \ll p^{1/2}, \]

and

\[ H(E_p) \leq \frac{p}{m} + 3 \]

by (6) and (7).

However, our method implies nontrivial bounds even if we substitute \( \omega_T \) in the definitions of \( D(E_T) \) and \( A(E_T) \) by any other primitive \( H \)th root of unity \( \omega_H \) with \( H \neq T \). (Note that we were able to apply the Weil bound since \( n \mapsto \omega^{-kn}_p \) is an additive character modulo \( p \).)

6.2 Character sequences modulo \( pq \)

In this section we give an example of a sequence with large correlation measure of order 4 but small autocorrelation, discrete Fourier transformation, ambiguity, and Hamming correlation.

For two primes \( p \neq q \) with \( m|\gcd(p-1, q-1) \) let \( \chi \) and \( \psi \) be multiplicative characters of order \( m \) modulo \( p \) and \( q \), respectively, with the convention \( \chi(0) = \psi(0) = 1 \). We study the \( pq \)-periodic sequence \( E_{pq} \) given by

\[ e_n = \chi(n)\psi(n), \quad n = 0, 1, \ldots \]

Then, considering the lags 0, \( p, q, p+q \) and the permutations

\[ \varphi_1(x) = \varphi_4(x) = x \quad \text{and} \quad \varphi_2(x) = \varphi_3(x) = \overline{x}, \quad x \in \mathcal{E}_m, \]

we get the following sum:

\[ \sum_{n=0}^{pq-1} \chi(n)\psi(n)\chi(n+p)\psi(n+p)\chi(n+q)\psi(n+q)\chi(n+p+q)\psi(n+p+q) = \sum_{n=0}^{pq-1} \chi(n)\psi(n)\chi(n+p)\psi(n+p)\chi(n+q)\psi(n+q)\chi(n+q)\psi(n+p) = pq \]

and we have

\[ \Gamma_4(E_{pq}) = pq. \]

However, we have

\[ C(E_{pq}) \ll \max\{p, q\} \]

since substituting

\[ n \equiv au + bv \mod pq \]

with unique \( 0 \leq u < p \) and \( 0 \leq v < q \), where

\[ au \equiv 1 \mod p \quad \text{and} \quad bp \equiv 1 \mod q, \]

we get

\[ \sum_{n=0}^{pq-1} \chi(n)\psi(n)\chi(n+t)\psi(n+t) \]

and

\[ \sum_{u=0}^{p-1} \chi(u)\psi(u+t) \]

\[ \sum_{v=0}^{q-1} \psi(v)\psi(v+t) \]
by the Chinese remainder theorem, which is \( \leq 9 \) if \( \gcd(t, pq) = 1 \), \( \ll p \) if \( \gcd(t, pq) = p \), and \( \ll q \) if \( \gcd(t, pq) = q \).

For the discrete Fourier transformation we get

\[
D(E_{pq}) \ll p^{1/2} q^{1/2}
\]

since

\[
\left| \sum_{n=0}^{pq-1} \chi(n) \psi(n) \omega_p^{-kn} \right| = \sum_{u=0}^{p-1} \chi(u) \omega_p^{-aku} \left| \sum_{v=0}^{q-1} \psi(v) \omega_q^{-bku} \right|.
\]

Similarly we get

\[
A(E_{pq}) \ll \max\{p^{1/2} q, pq^{1/2}\}
\]

and by (6)

\[
H(E_{pq}) \ll \frac{pq}{m} + \max\{p, q\}.
\]

Acknowledgment

We wish to thank the anonymous referees for their very careful reading.

References

APN functions of degree $4e$ with $e \equiv 3 \pmod{4}$

Florian Caullery

Received: 12/21/2012 / Accepted: 02/01/2013

Abstract In this paper, we show that there is no vectorial Boolean function of degree $4e$ with $e \equiv 3 \pmod{4}$ which is APN over infinitely many extensions of its field of definition. It is a new step in the proof of the conjecture of Aubry, McGuire and Rodier.

Keywords Vectorial Boolean function · Almost Perfect Non-linear functions · Algebraic surface · CCZ equivalence

1 Introduction

Block ciphers, such as the AES, correspond to a very important family of secret-key cryptosystems. The security of such systems partly relies on what is called the S-box. This is simply a vectorial Boolean function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ where $n$ is the size of the blocks. It is, in general, the only non linear operation in the algorithm.

One of the best known attacks on these systems is differential cryptanalysis. Nyberg proved in [18] that the S-boxes with the best resistance to such attacks are the ones which are said to be Almost Perfectly Non-linear (APN).

Let $q = 2^n$. A function $f : \mathbb{F}_q \to \mathbb{F}_q$ is said to be APN on $\mathbb{F}_q$ if the number of solutions in $\mathbb{F}_q$ of the equation

$$f(x+a) + f(x) = b$$

is at most 2 for all $a, b \in \mathbb{F}_q$, $a \neq 0$. The fact that $\mathbb{F}_q$ has characteristic 2 implies that the number of solutions is even for any function $f$ on $\mathbb{F}_q$.

The study of APN functions has focused on power functions and it was recently generalized to other functions, particularly polynomials (Carlet, Pott and al. [6, 11, 12]) or polynomials on small fields (Dillon [8]). On the other hand, several authors (Berger, Canteaut, Charpin, Laigle-Chapuy [2], Byrne, McGuire [5] or Jedlicka [3]) showed that APN functions did not exist in certain cases. Some also studied the notion of being APN on other fields than $\mathbb{F}_{2^n}$ (Leducq [16], Pott and al. [10, 19], Ness, Helleseth [17] or Wang, Zha [21, 22]).

Florian Caullery
Institut of Mathematiques of Luminy
Tel.: +336-24-912426
E-mail: florian.caullery@etu.univ-amu.fr
Toward a full classification of all APN functions, an approach is to show that certain polynomials are not APN for an infinity of extension of $F_2$.

Hernando and McGuire showed in [14] a result on classification of APN functions which was conjectured for 40 years: the only exponents such that the monomial $x^d$ is APN over an infinity of extension of $F_2$ are of the form $2^i + 1$ (Gold exponent) or $4^i - 2^i + 1$ (Kasami exponent). Those exponents are called exceptional exponents.

It led Aubry, McGuire and Rodier [1] to formulate the following conjecture:

**Conjecture:** (Aubry, McGuire and Rodier) A polynomial can be APN for an infinity of ground fields $F_q$ if and only if it is CCZ-equivalent (as defined by Carlet, Charpin and Zinoviev in [5]) to a monomial $x^d$ where $d$ is an exceptional exponent.

We define the functions which are APN for an infinity of ground fields $F_q$ as exceptional APN functions.

A way to prove this conjecture is to remark that being APN is equivalent to the fact that the rational points of a certain algebraic surface $X$ in a 3-dimensional space linked to the polynomial $f$ defining the Boolean function are all in a surface $V$ made of 3 planes and independent of $f$. We define the surface $X$ in the 3-dimensional affine space $A^3$ by

$$\phi(x, y, z) = f(x) + f(y) + f(z) + f(x + y + z)$$

which is a polynomial in $F_q[x, y, z]$. When the surface is irreducible or has an irreducible component defined over the field of definition of $f$, a Weil’s type bound may be used to approximate the number of rational points of this surface. When it is too large it means the surface is too big to be contained in the surface $V$ and the function $f$ cannot be APN.

This way enabled Rodier to prove in [20] that when the degree of $f$ is equal to $4e$ with $e \equiv 3 \pmod{4}$ and $\phi$ is not divisible by a certain form of polynomial then $f$ is not APN for an infinity of extensions of $F_q$. He also found all the APN functions of degree 12 and proved they are all CCZ-equivalent to $x^3$.

To continue in this way, we got interested in the exceptional APN functions of degree 20 which were the next ones on the list [7]. There were some differences between that case and the case of functions of degree 12. Nevertheless some common points appeared. In both cases, $\phi$ was divisible by the same form of polynomial. We also noticed that, in the case where $d = 12$, this polynomial was either, the $\phi$ associated to the cube of a $q$-affine permutation $L$ or the $\phi$ associated to $L(x^3)$. In addition we remarked that it forced $f$ to be a polynomial in $L(x)$. We made the assumption that it could be generalized when the degree of $f$ is $4e$ with $e \equiv 3 \pmod{4}$. It was a good hypothesis and we prove it in along this paper. This enabled us to state that if $f$ is an exceptional APN function it is CCZ-equivalent to a polynomial of odd degree. Using theorem 2.3 in [1] we could state that there is no polynomial of degree $4e$ with $e \equiv 3 \pmod{4}$, $e > 3$, which is APN over infinitely many extensions of $F_q$.

2 The state of the art

The best known APN functions are the Gold functions $x^{2^i+1}$ and the Kasami-Welch functions by $x^{d-2^i+1}$. These 2 functions are defined over $F_2$ and they are APN on any field $F_{2^n}$ if gcd $(m, i) = 1$. Aubry, McGuire and Rodier obtained the following results in [1].

**Theorem 1** (Aubry, McGuire and Rodier, [1]) If the degree of the polynomial function $f$ is odd and not an exceptional number then $f$ is not APN over $F_{q^n}$ for all $n$ sufficiently large.
Theorem 2 (Aubry, McGuire and Rodier [1]) If the degree of the polynomial function $f$ is $2e$ with $e$ odd and if $f$ contains a term of odd degree, then $f$ is not APN over $\mathbb{F}_q$, for all $n$ sufficiently large.

There are some results in the case of Gold degree $d = 2^i + 1$:

Theorem 3 (Aubry, McGuire and Rodier [1]) Suppose $f(x) = x^d + g(x)$ where $\deg(g) \leq 2^{i-1} + 1$. Let $g(x) = \sum_{j=0}^{2^{i-1}+1} a_j x^j$. Suppose moreover that there exists a nonzero coefficient $a_j$ of $g$ such that $\phi_j(x, y, z)$ is absolutely irreducible (where $\phi_j(x, y, z)$ denotes the polynomial $\phi_j(x, y, z)$ associated to $x^j$). Then $f$ is not APN over $\mathbb{F}_q$, for all $n$ sufficiently large.

This result has been consequently extended by Delgado and Janwa in [9] with the two following theorems:

Theorem 4 (Delgado and Janwa [9]) For $k \geq 2$, let $f(x) = x^{2^k+1} + h(x) \in \mathbb{F}_q$ where $\deg(h) \equiv 3 \pmod{4} < 2^i + 1$. Then $f$ is not an exceptional APN function.

and

Theorem 5 (Delgado and Janwa [9]) For $k \geq 2$, let $f(x) = x^{2^k+1} + h(x) \in \mathbb{F}_q$ where $\deg(h) \equiv 3 \pmod{4} < 2^i + 1$. If $\phi_2^k, \phi_d$ are relatively prime, then $f$ is not an exceptional APN function.

where

$$\phi_i = \frac{x^i + y^i + z^i + (x + y + z)^i}{(x+y)(y+z)(z+x)}$$

There also exist results for polynomials of Kasami degree $(d = 2^{2i} - 2^i + 1)$:

Theorem 6 (Férard, Oyono and Rodier [13]) Suppose $f(x) = x^d + g(x)$ where $d$ is a Kasami exponent and $\deg(g) \leq 2^{2i-1} - 2^{i-1} + 1$. Let $g(x) = \sum_{j=0}^{2^{2i-1} - 2^{i-1} + 1} a_j x^j$. Suppose moreover that there exist a nonzero coefficient $a_j$ of $g$ such that $\phi_j(x, y, z)$ is absolutely irreducible. Then $\phi(x, y, z)$ is absolutely irreducible.

Rodier proved the following results in [20]. We recall that for any function $f : \mathbb{F}_q \to \mathbb{F}_q$ we associate to $f$ the polynomial $\phi(x, y, z)$ defined by:

$$\phi(x, y, z) = \frac{f(x) + f(y) + f(z) + f(x + y + z)}{(x+y)(x+z)(y+z)}.$$

Theorem 7 (Rodier [20]) If the degree of a polynomial function $f$ is even such that $\deg(f) = 4e$ with $e \equiv 3 \pmod{4}$, and if the polynomials of the form

$$(x + y)(x + z)(y + z) + R$$

with

$$R(x, y, z) = c_1 (x^2 + y^2 + z^2) + c_4 (xy + xz + yz) + b_1 (x + y + z) + d,$$

for $c_1, c_4, b_1, d \in \mathbb{F}_q$, do not divide $\phi$ then $f$ is not APN over $\mathbb{F}_q^d$ for $n$ large.

There are more precise results for polynomials of degree 12.

Theorem 8 (Rodier [20]) If the degree of the polynomial $f$ defined over $\mathbb{F}_q$ is 12, then either $f$ is not APN over $\mathbb{F}_q^d$ for large $n$ or $f$ is CCZ equivalent to the Gold function $x^3$.
3 New Result

The goal of this paper is to prove the following theorem:

**Theorem 9** Let \( f : \mathbb{F}_q \to \mathbb{F}_q \) such that \( \deg(f) = 4e \) with \( e \equiv 3 \pmod{4} \) and \( e > 3 \), then \( f \) cannot be APN over infinitely many extensions of \( \mathbb{F}_q \).

4 Preliminaries

The following results are needed to prove theorem 9.

**Proposition 1** [20] The class of APN functions is invariant by adding a \( q \)-affine polynomial.

**Proposition 2** [20] The kernel of the map

\[
\phi(f) = \frac{f(x) + f(y) + f(z) + f(x+y+z)}{(x+y)(x+z)(y+z)}
\]

is made of \( q \)-affine polynomials.

We define the surface \( X \) in the 3-dimensional affine space \( \mathbb{A}^3 \) by

\[
\phi(x, y, z) = \frac{f(x) + f(y) + f(z) + f(x+y+z)}{(x+y)(x+z)(y+z)}
\]

and we call \( \bar{X} \) its projective closure.

**Proposition 3** [20] If the surface \( X \) has an irreducible component defined over the field of definition of \( f \) which is not one of the planes \((x+y)(x+z)(y+z) = 0\), the function \( f \) cannot be APN for infinitely many extensions of \( \mathbb{F}_q \).

**Lemma 1** [15] Let \( H \) be a projective hyper-surface. If \( \bar{X} \cap H \) has a reduced absolutely irreducible component defined over \( \mathbb{F}_q \) then \( \bar{X} \) has an absolutely irreducible component defined over \( \mathbb{F}_q \).

**Lemma 2** [1] Suppose \( d \) is even and write \( d = 2^j e \) where \( e \) is odd. In \( \bar{X} \cap H \) we have

\[
\phi_d = \phi_e (x, y, z)^{2^j} ((x+y)(x+z)(y+z))^{2^{j-1}}
\]

**Lemma 3** [20] The function \( x+y \) (and therefore \( A \)) does not divide \( \phi_i (x, y, z) \) for \( i \) an odd integer.

5 Proof of theorem 9

Let \( f : \mathbb{F}_q \to \mathbb{F}_q \) be a function of degree \( 4e \) where \( e \equiv 3 \pmod{4} \) which is APN over infinitely many extensions of \( \mathbb{F}_q \) and

\[
\phi_f(x, y, z) = \frac{f(x) + f(y) + f(z) + f(x+y+z)}{(x+y)(y+z)(z+x)}
\]

its associated polynomial.
Let us consider $\rho = Gal \left( \mathbb{F}_{q^3}/\mathbb{F}_q \right)$, $c_1, c_4, b_1, d \in \mathbb{F}_{q^3}$, $R(x, y, z) = c_1 \left(x^2 + y^2 + z^2 \right) + c_4 (xy + xz + yz) + b_1 (x + y + z) + d$ and $A = (x + y)(y + z)(z + x)$.

From theorem 5 and [20] we can assume that $(A + R) \left( A + \rho (R) \right) \left( A + \rho^2 (R) \right)$ divides the polynomial $\phi$ associated to $f$. As $(A + R) \left( A + \rho (R) \right) \left( A + \rho^2 (R) \right)$ is of total degree 9, we can write
\[
(A + R) \left( A + \rho (R) \right) \left( A + \rho^2 (R) \right) = \sum_{i=0}^{9} P_i,
\]
where $P_i$ is of degree $i$, we have
\[
\phi_f = \sum_{i=0}^{d} a_i \phi_i,
\]
with $a_i \in \mathbb{F}_q$. We fix $a_d$ to 1 as $\mathbb{F}_q$ is a field. As $\phi$ is of total degree $d - 3$ there exists a polynomial $Q \in \mathbb{F}_{q^3} [x, y, z]$ of total degree $d - 12$ such that
\[
Q = \sum_{i=0}^{d-12} Q_i,
\]
where $Q_i \in \mathbb{F}_{q^3} [x, y, z]$ is of total degree $i$ and
\[
\sum_{i=0}^{9} P_i \cdot \sum_{i=0}^{d-12} Q_i = \sum_{i=0}^{d} a_i \phi_i.
\]
As $\phi$ is a symmetrical polynomial in $x, y, z$ we can write it using symmetrical functions
\[
s_1 = x + y + z, \quad s_2 = xy + xz + yz \quad \text{and} \quad s_3 = xyz.
\]
Denoting $p_i = x^i + y^i + z^i$, we have $p_i = s_1 p_{i-1} + s_2 p_{i-2} + s_3 p_{i-3}$.

We remark that $\phi_1 = \frac{p_1 + p_2}{3}$ and that $A = (x + y)(y + z)(z + y) = s_1s_2 + s_3$.

The proof of theorem 9 is decomposed into two parts. The first one is to determine all the coefficients of $P$ identifying degree by degree with the $P_i$, $Q_i$ and $\phi_i$.

5.1 Determination of $R$

For all $k \in \{0, 1, \ldots, d\}$ we have
\[
a_k \phi_k = \sum_{i=0}^{q} P_i Q_{k-i-3}.
\]

**Proposition 4** If $R$ divides $\phi_f$, then $R = c_1 \phi_5 + c_1^3$ and the trace of $c_1$ in $\mathbb{F}_{q^3}$ is 0.

**Sketch of proof.**

We proceed by identification on the degree. For example, for degree $d - 3$, we have
\[
\phi_d = A^3 \phi_5^3 = P_9 Q_{d-12}.
\]

As $P_9 = A^3$, we obtain $Q_{d-12} = \phi_5^3$.

We continue in this way until degree $d - 5$ to obtain the expressions of all the $P_i$ and we deduce that $R$ is equal to $c_1 \phi_5 + c_1^3$. The reader will find the complete proof in the full paper. \qed
Let us consider \( L(x) = x(x + c_1)(x + \rho(c_1))(x + \rho^2(c_1)) \), since \( \text{tr}(c_1) = 0 \), \( L \) is a \( q \)-affine permutation. We have

\[
\frac{L(x)^3 + L(y)^3 + L(z)^3 + L(x + y + z)^3}{(x + y)(y + z)(z + x)} = (A + R)(A + \rho(R))(A + \rho^2(R)).
\]

So it means that the polynomial \( \phi \) associated to \( L(x)^3 \) divides \( \phi_f \).

The second part of the proof is to use this information in order to show that \( f \) is CCZ-equivalent to a polynomial of degree \( e \).

### 5.2 CCZ-equivalence

Let us consider \( c_1 \in \mathbb{F}_q \) such that \( \text{tr}(c_1) = 0 \) and \( R(x,y,z) = c_1 \phi_5 + c_1^3 \in \mathbb{F}_q[x,y,z] \). We recall that \( L(x) = x(x + c_1)(x + \rho(c_1))(x + \rho^2(c_1)) \).

**Theorem 10** Let \( f \) be a function such that \( \deg(f) = 4e \) with \( e \equiv 3 \pmod{4} \), \( e > 3 \), and such that the polynomials of the form

\[
(x+y)(x+z)(y+z) + R,
\]

divides \( \phi \), therefore \( f \) is CCZ-equivalent to \( x^e + S(x) \), where \( S \in \mathbb{F}_q[x] \) is of degree at most \( e - 1 \).

**Proof** Let us consider the set \( G \) of the polynomials of the form \( g(x) = L(x)^e + S(L(x)) \), where \( S \) is a polynomial of \( \mathbb{F}_q[x] \) of degree at most \( e - 1 \) with no monomials of exponent a power of 2. Let \( \delta \) be the number of power of 2 less or equal than \( e - 1 \). It is easy to remark that \( G \) defines an affine subspace of the vector space \( \mathbb{F}_q[x] \) of dimension \( e - \delta \). We denote by \( \phi_f \) the polynomial \( \phi \) associated to \( g \) and \( \phi_{L^n} \) the polynomial \( \phi \) associated to \( L^n \). So we have

\[
\phi_f = \phi_{L^n} + S(\phi_{L^n}).
\]

Now let us consider the set \( F \) of all the polynomials \( f \) of degree \( 4e \) with leading coefficient 1 such that \( \phi_f \) divides their associated polynomials \( \phi \) and such that \( f \) does not have any monomial of exponent a power of 2. The goal of this proof is to show that \( G = F \). We begin by proving that \( F \subseteq G \), then we show that they have the same dimension.

**Lemma 4** The set \( G \) is a subset of \( F \).

**Proof** It is sufficient to prove that \( \phi_{L^n} \) divides \( \phi_f \) for all \( n \geq 3 \).

We know that \( x^3 + y^3 + z^3 + (x + y + z)^3 = A \) divides \( x^a + y^a + z^a + (x + y + z)^a \). Putting

\[
X = L(x)
\]
\[
Y = L(y)
\]
\[
Z = L(z)
\]
we have \( X^3 + Y^3 + Z^3 + (X + Y + Z)^3 \) divides \( X^n + Y^n + Z^n + (X + Y + Z)^n \). As \( \text{tr}(c_1) = 0 \), \( L(x) \) is a linearized polynomial so \( X + Y + Z = L(x) + L(y) + L(z) = L(x + y + z) \) therefore \( L(x)^3 + L(y)^3 + L(z)^3 + L(x + y + z)^3 \) divides \( L(x)^{ae} + L(y)^{ae} + L(z)^{ae} + L(x + y + z)^{ae} \) then \( \phi_{L^n} \) divides \( \phi_f \).
Lemma 5 F defines an affine subspace of the vector space $\mathbb{F}_q[x]$ of dimension less or equal than $e - \delta$.

Proof We consider the mapping:

$$\phi : F \rightarrow \mathbb{F}_q^{e-\delta}$$

$$f \rightarrow (a_{d-4}, \ldots, a_{12})$$

It is sufficient to prove that this mapping is one-to-one.

Let $f$ and $f'$ in $F$ be two elements such that $\phi (f) = \phi (f')$. We write $f = \sum_{i=0}^{d} a_i x^i$ and $f' = \sum_{i=0}^{d} a'_i x^i$. We note $a_i \phi = \sum_{j=1}^{9} P_j Q_{k-1, j-3}$ and $a'_i \phi = \sum_{j=0}^{9} P'_j Q'_{k-1, j-3}$.

We will show by induction that $a_i = a'_i$ for all $0 \leq i \leq d$ and that $Q_j = Q'_j$ for all $0 \leq i \leq d - 12$.

We have $a_d = a'_d = 1$ and $Q_{d-12} = Q'_{d-12} = \phi^d$.  

Suppose that $a_i = a'_i$ and that $Q_{j-12} = Q'_{j-12}$ for $j > i$. Let us show that $a_i = a'_i$ and $Q_{j-12} = Q'_{j-12}$ if $4$ does not divide $i$.

If $i \geq 12$, we have

$$a_i \phi = \sum_{\sup(0,j-d+9)} P_k Q_{i-k-3}^i = A^3 Q_{i-12}^i + \sum_{\sup(0,j-d+9)} P_k Q_{i-k-3}^i,$$

so $A^3$ divides

$$a_i \phi + \sum_{\sup(0,j-d+9)} P_k Q_{i-k-3}^i.$$

It divides

$$a'_i \phi + \sum_{\sup(0,j-d+9)} P_k Q'_{i-k-3}^i = a'_i \phi + \sum_{\sup(0,j-d+9)} P_k Q_{i-k-3}^i,$$

because $i - k - 3 \geq i - 11$. So it divides $(a_i + a'_i) \phi$. If $4$ does not divide $i$ then $A^3$ does not divide $\phi$, so $a_i = a'_i$ and

$$Q_{i-12} = \frac{a_i \phi + \sum_{\sup(0,j-d+9)} P_k Q_{i-k-3}^i}{A^3} = \frac{a'_i \phi + \sum_{\sup(0,j-d+9)} P_k Q'_{i-k-3}^i}{A^3} = Q'_{d-12}.$$

From lemma 4 and 5 we have $F = G$.

So every $f \in F$ is of the form $L(x)^e + S(L(x))$ so they are CCZ equivalent to $x^e + S(x)$. If $f$ is of degree $4e$ with leading coefficient $1$ such that $\phi_3$ divides their associated polynomials $\phi$ and has monomials of exponent a power of $2$, then $f$ is CCZ-equivalent to a polynomial in $F$ therefore it is also CCZ-equivalent to $x^e + S(x)$.

We now have that $f$ is CCZ-equivalent to a polynomial of degree $e$ which is odd. As $e \equiv 3 \mod 4$ and $e > 3$, $e$ cannot be a Gold or a Kasami exponent. Therefore from theorem, $1 f$ cannot be APN.

Acknowledgements This work has been done with Francois Rodier as adviser. I would like to thank him for all his valuable advices and his infinite patience.
References

4. L. Budaghyan and C. Carlet and P. Felke and G. Leander An infinite class of quadratic APN functions which are not equivalent to power mappings, Cryptology ePrint Archive, n° 2005/359.
7. F. Caullery, Polynomial functions of degree 20 which are not APN infinitely often. eprint arXiv:1212.4638.
15. G. Leander and F. Rodier Bounds on the degree of APN Polynomials. The case of $x^{-1} + g(x)$. Designs, Codes and cryptography, 0925-1022, 2009.
A matrix approach for constructing quadratic APN functions

Yuyin Yu · Mingsheng Wang · Yongqiang Li

Received: date / Accepted: date

Abstract We find a correspondence between quadratic APN functions and a special kind of matrices. Based on this result, we have developed efficient algorithms to construct quadratic APN functions. Up to now, we have found more than 470 classes of new CCZ-inequivalent quadratic APN functions on \(\mathbb{F}_{2^7}\), which is 20 times more than known ones. And we also have found more than 1000 classes of new CCZ-inequivalent quadratic APN function on \(\mathbb{F}_{2^8}\).

Keywords APN · quadratic functions · EA-equivalence · CCZ-equivalence

1 Introduction

Permutations with low differentially uniform are useful in cryptography. For example, the Advanced Encryption Standard (AES) [11] chooses a differentially 4 uniform permutation as its S-box. In 2009, Dillon et al. [3,12,4] found an APN [20] permutation in dimension six, which is the first APN permutation in even dimension. Their idea can be summarized as checking whether there is a permutation CCZ-equivalent to the given APN function. Thus, if we want to find new APN permutations in even dimensions, we must find new APN functions first.

An APN function is new if it is CCZ-inequivalent to any known ones. For a long time, finding new APN functions is an important topic in cryptography. In recent years, most of the new found APN functions are quadratic. Related work appeared in [1]-[8], [12]-[17], [23]. For a systematic knowledge of APN functions, the readers can turn to [9].

Our aim is to find as many new APN functions as possible, especially on \(\mathbb{F}_{2^8}\). Then, we will check whether these new APN functions are CCZ-equivalent to some...
permutations. If some function is CCZ-equivalent to a permutation, then there
must exist an APN permutation.

Our idea results from the following lemma, which is Theorem 2.3 in [19].

Lemma 1 [19] Let \( \{ \theta_1, \theta_2, \ldots, \theta_n \} \) be any given basis of \( \mathbb{F}_{2^n} \) over \( \mathbb{F}_2 \), and let \( L(x) \) be linearized polynomial over \( \mathbb{F}_{2^n} \). Then there exists a unique vector \( (\beta_1, \beta_2, \ldots, \beta_n) \in \mathbb{F}_{2^n}^n \) such that

\[
L(x) = \sum_{j=1}^{n} \text{Tr}(\theta_j x) \beta_j = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} \beta_j \theta_j^{2^i-1} \right) x^{2^i-1}.
\]

Moreover, let \( k \) be an integer such that \( 0 \leq k \leq n \), then \( \dim_{\mathbb{F}_2}(\ker(L)) = k \) if and only if \( \text{Rank}_{\mathbb{F}_2}(\beta_1, \beta_2, \ldots, \beta_n) = n - k \).

Based on this lemma, we can build a correspondence between quadratic APN functions and a special kind of matrix. Using the nice structure of such matrices we can design some efficient algorithms to construct quadratic APN functions. Our main contributions can be concluded as follows.

We find one to one correspondence between restricted quadratic APN functions and QAMs, and we also list many properties of the QAMs without proof in this paper. Based on these properties, we have designed some efficient algorithms to search the new APN functions. Up to now, we have found more than 470 new CCZ-in-equivalent APN functions on \( \mathbb{F}_{2^7} \), and more than 1000 new CCZ-in-equivalent quadratic APN functions on \( \mathbb{F}_{2^8} \). The number of CCZ-in-equivalent quadratic APN functions on \( \mathbb{F}_{2^8} \) is still increasing quickly. We have checked all these new APN functions on \( \mathbb{F}_{2^8} \), none of them is CCZ-equivalent to a permutation.

Our work is just a beginning, we think that more useful results will be obtained in the future.

2 Notation

Let \( n \) be a positive integer, \( \mathbb{F}_{2^n} \) be a finite fields with \( 2^n \) elements, and \( \mathbb{F}_{2^n}[x] \) be the ring of polynomial in variable \( x \).

Definition 1 A mapping \( F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n} \) is called differentially \( \delta(F) \) uniform if

\[
\delta(F) = \max_{a \in \mathbb{F}_{2^n}, b \in \mathbb{F}_{2^n}} \# \Delta_F(a, b),
\]

where \( \Delta_F(a, b) = \{ x \in \mathbb{F}_{2^n} : F(x+a)+F(x) = b \} \) and \( \# \Delta_F(a, b) \) is the cardinality of the set \( \Delta_F(a, b) \). If \( \delta(F) = 2 \), we can say that \( F \) is APN (Almost perfect nonlinear).

Definition 2 Suppose \( F \) and \( F' \) are two functions from \( \mathbb{F}_{2^n} \) to \( \mathbb{F}_{2^n} \). Then

1) \( F \) and \( F' \) are EA-equivalent (Extended affine equivalent) if

\[
F'(x) = A_1(F(A_2(x))) + A_3(x),
\]

where \( A_1 \) and \( A_2 \) are affine permutations on \( \mathbb{F}_{2^n} \), and \( A_3 \) is an affine function on \( \mathbb{F}_{2^n} \).

2) \( F \) and \( F' \) are CCZ-equivalent (Carlet-Charpin-Zinoviev equivalent) if there is an affine permutation which maps \( G_F \) to \( G_{F'} \), where \( G_F = \{ (x, F(x)) : x \in \mathbb{F}_{2^n} \} \) is the graph of \( F \), and \( G_{F'} \) is the graph of \( F' \).
According to Yoshiara’s results [21], two quadratic APN functions are CCZ-equivalent if and only if they are EA-equivalent.

**Definition 3** Quadratic functions without linear or constant terms are called restricted quadratic functions in this paper.

**Definition 4** [18] Two bases \( \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \) and \( \{\theta_1, \theta_2, \ldots, \theta_n\} \) of \( \mathbb{F}_2^n \) over \( \mathbb{F}_2 \) are said to be dual basis if for \( 1 \leq u, j \leq n \) we have

\[
\text{Tr}(\alpha_u \theta_j) = \begin{cases} 
0 & \text{for } u \neq j, \\
1 & \text{for } u = j.
\end{cases}
\]

We will use the following convention and notation throughout the paper.

(i) For positive numbers \( r, s, \mathbb{F}_2^{r \times s} \) denote the space of \( r \times s \) matrices over \( \mathbb{F}_2 \). For a matrix \( A \), \( A[i] \) denote the \( i \)th row of \( A \), and \( A[i,j] \) the \((i,j)\) element of \( A \). Moreover, \( B = \text{Submatrix}(A,1,1,r,c) \) denotes the \( r \times c \) submatrix of \( A \) which consists of the first \( r \) rows and the first \( c \) columns.

(ii) Suppose \( \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \) is a basis of \( \mathbb{F}_2^n \) over \( \mathbb{F}_2 \), and \( \{\theta_1, \theta_2, \ldots, \theta_n\} \) is its dual basis. Let \( M_{\alpha} \in \mathbb{F}_2^{n \times n} \) and \( M_{\theta} \in \mathbb{F}_2^{n \times n} \) with \( M_{\alpha}[i,u] = \alpha_u^\theta \) and \( M_{\theta}[i,u] = \theta_u^\alpha \) for \( 1 \leq u, i \leq n \). Then \( M_{\alpha}^t M_{\theta} = (\text{Tr}(\alpha_u \theta_j))_{n \times n} \) for \( 1 \leq u, j \leq n \). So \( M_{\alpha}^t M_{\theta} = I_n \), where \( I_n \) is the identity matrix of size \( n \). Thus \( M_{\theta} = M_{\alpha}^{-1} \) (\( M_{\alpha}^{t} \) is the transpose of \( M_{\alpha} \)).

(iii) Let \( \eta_1, \eta_2, \ldots, \eta_m \) be \( m \) elements on \( \mathbb{F}_2^n \) \((m,n \geq 1)\), and \( B = (\eta_1, \eta_2, \ldots, \eta_m) \in \mathbb{F}_2^m \). \( \text{Span}(B) = \text{Span}(\eta_1, \eta_2, \ldots, \eta_m) \) denotes the subspace spanned by \( \{\eta_1, \eta_2, \ldots, \eta_m\} \) over \( \mathbb{F}_2 \). \( \text{Rank}_{\mathbb{F}_2}(\eta_1, \eta_2, \ldots, \eta_m) \) is the dimension of \( \text{Span}(B) \) over \( \mathbb{F}_2 \), which we call the rank of \( B \) (over \( \mathbb{F}_2 \)). Suppose \( \eta_i = \sum_{j=1}^{m} \lambda_{i,j} \eta_j \) for \( 1 \leq i \leq m \), where \( \lambda_{i,j} \in \mathbb{F}_2 \) for all \( i, j \). Define a \( m \times n \) matrix \( A = (\lambda_{i,j})_{m \times n} \). Then \( \text{Rank}_{\mathbb{F}_2}(\eta_1, \eta_2, \ldots, \eta_m) \) equals the rank of \( A \).

**Definition 5** Let \( H = (h_{u,v})_{n \times n} \) be a \( n \times n \) matrix defined on \( \mathbb{F}_2^n \). Then the matrix \( H \) is called a QAM (quadratic APN matrix) if
1. \( H \) is symmetric and the elements in its main diagonal are all zeros;
2. Every nonzero linear combination of the \( n \) rows (or “columns” since \( H \) is symmetric) of \( H \) has rank \( n - 1 \).

### 3 The Correspondence between Restricted Quadratic APN Function and QAMs

Let \( F(x) = \sum_{1 \leq i \leq n} c_{i,t} x^{2^{t-1} + 2^{i-1}} \in \mathbb{F}_2^n[x] \) be a restricted quadratic function, and \( C_F \) be the \( n \times n \) matrix such that \( C_F[i,t] = C_F[i,t] = c_{i,t} \) for \( 1 \leq t < i \leq n \) and \( C_F[i,i] = 0 \) for \( 1 \leq i \leq n \).

Given a basis \( \alpha = \{\alpha_1, \ldots, \alpha_n\} \) for \( \mathbb{F}_2^n \) over \( \mathbb{F}_2 \), and let \( M = M_{\alpha}, \Theta = M_{\theta} \). For any restricted quadratic function \( F(x) \), let \( H = M^t C_F M \). Then \( H \) is a symmetric matrix over \( \mathbb{F}_2^n \) with main diagonal elements zeros.

Conversely, for a symmetric matrix \( H \) with main diagonal elements all zeros, we can define a unique restricted quadratic function \( F(x) \) such that \( H = M^t C_F M \). \( F(x) \) is called the quadratic function defined by \( H \) relative to the ordered basis \( \alpha \).

Based on Lemma 1, we can build a one to one correspondence between restricted quadratic APN functions and QAMs.
Theorem 1 Let $F(x) = \sum_{1 \leq i < j \leq n} c_{i,j}x^{2^{i-1}+2^{j-1}} \in \mathbb{F}_{2^n}[x]$, $C_F$ and $M$ be defined as above, and $H = M^t C_F M$. Then, $\delta(F) \leq 2^k$ if and only if any nonzero linear combination of the $n$ rows of $H$ has rank at least $n-k$. In particular, $F$ is APN on $\mathbb{F}_{2^n}$ if and only if $H$ is a QAM.

Proof Let $D_a(x) = F(x+a) + F(x) + F(a)$. Note that $D_a(x)$ is a linear polynomial on $x$, so we have

$$\delta(F) = 2^k \text{ if and only if } \max\{\dim_{\mathbb{F}_2}(\text{Ker}(D_a)) \mid a \in \mathbb{F}_{2^n}^n\} = k. \quad (1)$$

Expanding $D_a(x)$ we can get

$$D_a(x) = \sum_{j=1}^{n} \sum_{t=1}^{n} c_{j,t}a^{2^{t-1}} \in \mathbb{F}_{2^n}[x]. \quad (2)$$

$$\{\theta_1, \theta_2, \ldots, \theta_n\}$$ is the given basis of $\mathbb{F}_{2^n}$, so according to Lemma 1, there exists a unique vector $(\beta_1(a), \beta_2(a), \ldots, \beta_n(a)) \in \mathbb{F}_{2^n}$ such that

$$D_a(x) = \sum_{j=1}^{n} \text{Tr}(\theta_j x) \beta_j(a) = \sum_{j=1}^{n} \sum_{i=1}^{n} \beta_j(a) a^{2^{i-1}} x^{2^{i-1}}, \quad (3)$$

where $\beta_j(a)$ ($1 \leq j \leq n$) are some linear polynomials on $a$.

By Lemma 1 and (3), we have

$$\dim_{\mathbb{F}_2}(\text{Ker}(D_a)) = k \iff \text{Rank}_{\mathbb{F}_2} \{\beta_1(a), \beta_2(a), \ldots, \beta_n(a)\} = n - k. \quad (4)$$

Hence by (1) and (4), we have

$$\delta(F) = 2^k \iff \min_{a \in \mathbb{F}_{2^n}^n} \{\text{Rank}_{\mathbb{F}_2} \{\beta_1(a), \beta_2(a), \ldots, \beta_n(a)\}\} = n - k. \quad (5)$$

Comparing (2) with (3), it can be concluded that

$$C_F \begin{pmatrix} a^{2^n} \\ a^{2^{1}} \\ \vdots \\ a^{2^{n-1}} \end{pmatrix} = \Theta \begin{pmatrix} \beta_1(a) \\ \beta_2(a) \\ \vdots \\ \beta_n(a) \end{pmatrix},$$

so

$$\begin{pmatrix} \beta_1(a) \\ \beta_2(a) \\ \vdots \\ \beta_n(a) \end{pmatrix} = \Theta^{-1} C_F \begin{pmatrix} a^{2^n} \\ a^{2^{1}} \\ \vdots \\ a^{2^{n-1}} \end{pmatrix} = M^t C_F \begin{pmatrix} a^{2^n} \\ a^{2^{1}} \\ \vdots \\ a^{2^{n-1}} \end{pmatrix}. \quad (6)$$

Let $a = a_1 \alpha_1 + a_2 \alpha_2 + \cdots + a_n \alpha_n$, where $a_i \in \mathbb{F}_2$ for all $1 \leq i \leq n$, thus we have

$$\begin{pmatrix} a^{2^n} \\ a^{2^{1}} \\ \vdots \\ a^{2^{n-1}} \end{pmatrix} = \begin{pmatrix} \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{2^{n-1}} & \alpha_2^{2^{n-1}} & \cdots & \alpha_n^{2^{n-1}} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = M \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$
Based on the above results, (6) is equal to

\[
\begin{pmatrix}
\beta_1(a) \\
\beta_2(a) \\
\vdots \\
\beta_n(a)
\end{pmatrix} = M^t C_F M \begin{pmatrix}
a_1 \\
a_2 \\
\vdots \\
a_n
\end{pmatrix} = H \begin{pmatrix}
a_1 \\
a_2 \\
\vdots \\
a_n
\end{pmatrix}.
\]

(7)

The rightmost expression of (7) can be understood as a nonzero linear combination of the columns of \( H \), so, based on (5) and (7) we have

\[
\delta(F) \leq 2^k \iff \{ \text{any nonzero linear combination of the columns of } H \text{ has rank at least } n - k \}. \quad (8)
\]

Note that \( D_a(x) = D_a(x + a) \). So \( \delta(F) \) must be even, that is, \( \delta(F) = 1 \) will never happen. Since \( H \) is symmetric, we have

\[
\delta(F) = 2 \iff \{ \text{any nonzero linear combination of the rows of } H \text{ has rank at least } n - 1 \}. \quad (9)
\]

\[\square\]

The matrix \( H \) which is associated with \( F(x) \) in Theorem 1 is called the matrix of \( F(x) \) relative to the ordered basis \( \{\alpha_1, \ldots, \alpha_n\} \).

Note that \( M \) is an invertible matrix on \( \mathbb{F}_2^n \), so the correspondence between restricted quadratic APN functions and QAMs is one to one. It seems that another similar approach have been considered by Knuth and Edel, the readers can turn to Edel’s ppt [13] for details.

Theorem 1 is very useful when we want to study the differential properties of the quadratic functions, and it can be generalized to \( F_{p^n} \), where \( p \) is any prime and \( n \) is a positive integer. In this paper, we use only this theorem to study the quadratic APN functions on \( \mathbb{F}_{2^n} \).

4 Properties of QAMs

We will introduce some properties of the QAMs in this section. Let \( F(x) \) be a given restricted quadratic function. First we should like to inquire what happens to corresponding matrices when the ordered basis is changed. Let \( \alpha = \{\alpha_1, \ldots, \alpha_n\} \) and \( \beta = \{\beta_1, \ldots, \beta_n\} \) be two ordered bases for \( \mathbb{F}_{2^n} \) over \( \mathbb{F}_2 \). Assume \( H_\alpha \) and \( H_\beta \) are corresponding matrices for \( F(x) \) relative to the \( \alpha, \beta \), respectively. How are the matrices \( H_\alpha \) and \( H_\beta \) related?

As we know, there is a unique invertible \( n \times n \) matrix \( P \) such that

\[
(\beta_1, \ldots, \beta_n) = (\alpha_1, \ldots, \alpha_n)P.
\]

Hence we have \( M_\beta = M_\alpha P \). So we have \( H_\beta = M_\beta^t C_F M_\beta = P^t H_\alpha P \).

Conversely, assume that \( H', H \) are two \( n \times n \) symmetric matrices with main diagonal elements all zeros such that \( H' = P^t HP \) for an invertible matrix \( P \) over \( \mathbb{F}_2 \). Let \( F(x) \) be the quadratic function defined by \( H \) relative to the ordered basis \( \alpha \). Let \( \gamma = \{\gamma_1, \ldots, \gamma_n\} \) be defined by \( (\gamma_1, \ldots, \gamma_n) = (\alpha_1, \ldots, \alpha_n)P \). Then \( \gamma \) is a basis for \( \mathbb{F}_{2^n} \), and \( F(x) \) is also the quadratic function defined by \( H' \) relative to
ordered basis \( \gamma \). Now let \( F'(x) \) be the quadratic function defined by \( H' \) relative to \( \alpha \), then how are the functions \( F(x) \) and \( F'(x) \) related?

We use the following two theorems to explain this problem and another similar problem.

**Theorem 2** Let \( H \in \mathbb{F}_2^{n \times n} \) be a symmetric matrix with main diagonal elements all zeros, and \( P \in \mathbb{F}_2^{n \times n} \) be an invertible matrix. Suppose \( H' = P^t HP \), then the quadratic functions defined by \( H \) and \( H' \) relative to an ordered basis \( \alpha \) are EA-equivalent. Especially, \( H \) is a QAM if and only if \( H' \) is a QAM.

**Theorem 3** Let \( H = (h_{u,v}) \in \mathbb{F}_2^{n \times n} \) be a symmetric matrix with main diagonal elements all zeros, and \( L \) be a linear permutation on \( \mathbb{F}_2^m \). Let \( H' = (h'_{u,v}) \in \mathbb{F}_2^{n \times n} \) such that \( h'_{u,v} = L(h_{u,v}) \) for all \( 1 \leq u, v \leq n \). Then the quadratic functions defined by \( H \) and \( H' \) relative to \( \alpha \) are EA-equivalent. In particular, \( H \) is a QAM if and only if \( H' \) is a QAM.

We get the following results from our experiments, which prove the correctness of Algorithm 1.

**Lemma 2** Let \( H \in \mathbb{F}_2^{m \times n} \) be a symmetric matrix with main diagonal elements all zeros. Then every nonzero linear combination of the \( n \) rows of \( H \) has rank at most \( n - 1 \).

**Definition 6** Let \( H \in \mathbb{F}_2^{m \times k} \) \((m, k \leq n)\). \( H \) is called proper if every nonzero linear combination of the \( m \) rows of \( H \) has rank at least \( k - 1 \).

**Theorem 4** Let \( A = (a_{i,j}) \in \mathbb{F}_2^{r \times c} \) \((1 \leq r < c \leq n)\) such that \( a_{i,j} = a_{j,i} \) and \( a_{i,i} = 0 \) for \( 1 \leq i, j \leq r \). Let \( A[\cdot, k] \) be the \( k \)-th column of \( A \), and \( b = \sum_{k=1}^{c} \lambda_k A[\cdot, k] \), where \( \lambda_k \in \mathbb{F}_2 \) for \( 1 \leq k \leq c \). Assume \( t = \text{Rank}_{\mathbb{F}_2} \{b[1], b[2], \ldots, b[r]\} \). If \( A \) is proper, then we have:

i) If \( (\lambda_{r+1}, \ldots, \lambda_c) = 0 \), then \( t = r - 1 \);

ii) If \( (\lambda_{r+1}, \ldots, \lambda_c) \neq 0 \), then \( t = r \).

According to Theorem 4, we get the following corollary.

**Corollary 1** Let \( H = (h_{u,v})_{n \times n} \) be an \( n \times n \) symmetric matrix over \( \mathbb{F}_2^m \), and \( A = \text{Submatrix}(H, 1, 1, r, c) \). Suppose \( B = A^t = \text{Submatrix}(H, 1, 1, c, r) \). Then \( A \) is proper implies that \( B \) is also proper.

Corollary 1 is useful in our algorithm for constructing QAMs. Our algorithm can be summarized as “Guess and determine”. The basic idea is: every submatrix of a QAM must be proper (See Definition 6). So, if a matrix has a submatrix which is not proper, it cannot be a QAM. Based on this corollary, we can exclude some improper candidates in advance when we haven’t known the whole values of the matrix. We know that every QAM is symmetric. So we will know the values of \( A = \text{Submatrix}(H, 1, 1, r, c) \) and \( B = A^t = \text{Submatrix}(H, 1, 1, c, r) \) at the same time, but according to Corollary 1. We only need to check whether \( A \) is proper. Thus we can avoid some unnecessary checking in our searching algorithm.
5 An Algorithm for Constructing QAMs

5.1 A Problem and the Algorithm

In this section, we introduce a problem and next we describe how to construct QAMs through solving this problem.

**Problem 1** Let $e_i$ be a vector of length $n$ with $e_i[i] = 1$ and $e_i[j] = 0$ for $i \neq j$. The problem is, how to find $\mathcal{P} = (x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^{n-1}$ which satisfies

$$
\lambda_1 x_1 + \cdots + \lambda_{n-1} x_{n-1} \in S_{\lambda_1 e_1 + \cdots + \lambda_{n-1} e_{n-1}},
$$

for all $(\lambda_1, \ldots, \lambda_{n-1}) \in \mathbb{F}_2^{n-1}\backslash \{0\}$, where $S_{\lambda_1 e_1 + \cdots + \lambda_{n-1} e_{n-1}}$ are some subsets of $\mathbb{F}_2^n$.

(10) consists of $2^{n-1} - 1$ conditions, we need to find all the qualified $\mathcal{P}$. As a matter of fact, Problem 1 is the core part for constructing QAMs.

Given an $n \times n$ QAM matrix $H$ over $\mathbb{F}_2^n$, we wish to reassign the values of the last column of $H$ to get some new QAMs. Let $A = \text{Submatrix}(H, 1, 1, n-1, n-1)$, it is easy to see that $A$ is proper. By Lemma 2, any nonzero linear combination of the $n-1$ rows of $A$ has rank $n-2$.

Let $c = (x_1, \ldots, x_{n-1})^t$, and $H' = \left( \begin{array}{cc} A & c \\ c^t & 0 \end{array} \right)$. We want to choose suitable $c$ to make $H'$ a QAM. Actually, by Theorem 4 (ii), we need only to choose $c = (\cdots, x_{n-1})^t$ to satisfy (10), where $S_{\lambda_1 e_1 + \cdots + \lambda_{n-1} e_{n-1}} = \mathbb{F}_2^n \backslash \text{Span}(\lambda_1 A[1] + \cdots + \lambda_{n-1} A[n-1])$.

We can shrink $S_{e_1}$ in (10). Let $V = \text{Span}(A[1], A[1, 2], \ldots, A[1, n-1])$, in (10), $S_{e_1} = \mathbb{F}_2^n \backslash V$, which equals to $(V + a_1) \cup (V + a_2) \cup (V + a_3)$ for some $a_i \in \mathbb{F}_2^n$, $1 \leq i \leq 3$ because of $\dim(V) = n-2$. Since $x_1 \in S_{e_1}$, there exists $y \in V$ such that $x_1 = y + a_i$ for some $i$, i.e., $x_1 = y + a_i$. Since $y \in V$ and $A[1, 1] = 0$, $y = \lambda_2 A[1, 2] + \cdots + \lambda_{n-1} A[1, n-1]$ for some $\lambda_i \in \mathbb{F}_2$, $i = 2, \ldots, n-1$. So we may perform suitable column transformations to change $x_1$ into $a_i$, and perform the corresponding row transformations to change $H'[n, 1]$ into $a_i$. Since we consider only to find CCZ-inequivalent functions, so by Theorem 2, we may take $S_{e_1} = \{a_1, a_2, a_3\}$. Because in the above transformation, we do not use the first column, therefore, based on the same reason as the $S_{e_1}$, we may take $S_{e_2} = \{b_1, \ldots, b_l\}$, where $l = 2^{n-1} - 2^{n-3}$.

Further, given a QAM $H$, we may also reassign the values of the last two columns of $H$ to get some new QAMs. This can also be reduced to Problem 1, the difference is that we must apply the problem twice. Similarly, we can reassign more columns of $H$, so this method can generate all CCZ-inequivalent quadratic APN functions if we change enough columns.

In view of the above discussions, an algorithm for solving problem 1 is important for our approach for constructing new quadratic functions. In the following, we describe an algorithm for solving Problem 1.

**Algorithm 1** Step 1. Initialization. Given a QAM $H$ over $\mathbb{F}_2^n$, let $A = \text{Submatrix}(H, 1, 1, n-1, n-1)$. Let $e_t \in \mathbb{F}_2^{n-1}$ with $e_t[t] = 1$, and $e_t[j] = 0$ for $j \neq t$. Let $S_{\lambda_1, \ldots, \lambda_{n-1}}^{(1)} = \mathbb{F}_2^n \backslash \text{Span}(\lambda_1 A[1] + \cdots + \lambda_{n-1} A[n-1])$ for all $(\lambda_1, \ldots, \lambda_{n-1}) \in \mathbb{F}_2^{n-1}\backslash \{0\}$. Let $i = 1$. 


Step 2. For each \( x_i \in S_{\lambda}^i \), do Step 3.

Step 3. If \( i = n - 1 \), then do Step 5, else do Step 4.

Step 4. Let \( H[i, n] = H[n, i] = x_i \). For all \((\lambda_{i+1}, \ldots, \lambda_{n-1}) \in F_2^{n-1-i} \setminus \{0\}\), define \( S_{\lambda}^{i+1} = S_{\lambda}^i \cap S_{\lambda_{i+1}} \), where \( \lambda = (0, \ldots, 0, \lambda_{i+1}, \ldots, \lambda_{n-1}) \in F_2^{n-1} \). Then let \( i := i + 1 \), turn to Step 2.

Step 5. Let \( H[n - 1, n] = H[n, n - 1] = x_{n-1} \), then output \( H \).

5.2 An Example

Algorithm 1 is the core part of our program, if we want to find new APN functions on \( F_2^n \) for \( n \geq 8 \), then we must change the values of a QAM for at least two columns (and rows). We give an example in the following.

It is well-known that \( x^3 \) is a restricted quadratic APN function on \( F_2^n \). Let \( n = 8 \), \( g \) be the default primitive element used in Magma, and \( C \) be an \( 8 \times 8 \) matrix such that \( C[1, 2] = C[2, 1] = 1 \) and \( C[i, i] = 0 \) for all the other values. Suppose the \( M \) defined in Definition 4 is an \( 8 \times 8 \) matrix such that \( M[i, j] = (g^{11})^{2^{i-1} + 2^{j-1}} \) for \( 1 \leq i, j \leq n \) (Note that \( \{ (g^{11}), (g^{11})^2, \ldots, (g^{11})^{2^{n-1}} \} \) is a basis of \( F_2^n \) over \( F_2 \)). Then we can get the corresponding QAM of \( x^3 \):

\[
H = M^t CM = \begin{pmatrix}
0 & g^{34} & g^{81} & g^{83} & g^{170} & g^{106} & g^{84} & g^{17} \\
g^{34} & 0 & g^{68} & g^{162} & g^{166} & g^{85} & g^{212} & g^{168} \\
g^{81} & g^{68} & 0 & g^{136} & g^{69} & g^{77} & g^{170} & g^{169} \\
g^{83} & g^{162} & g^{136} & 0 & g^{17} & g^{138} & g^{154} & g^{85} \\
g^{170} & g^{166} & g^{69} & g^{17} & 0 & g^{34} & g^{21} & g^{13} \\
g^{106} & g^{85} & g^{77} & g^{138} & g^{34} & 0 & g^{88} & g^{12} \\
g^{84} & g^{212} & g^{170} & g^{154} & g^{21} & g^{68} & 0 & g^{136} \\
g^{17} & g^{168} & g^{169} & g^{85} & g^{53} & g^{42} & g^{136} & 0
\end{pmatrix}
\]

Reassign the values of the last two columns (and rows), we can get a new QAM

\[
H' = \begin{pmatrix}
0 & g^{34} & g^{81} & g^{83} & g^{170} & g^{106} & g^{84} & 1 \\
g^{34} & 0 & g^{68} & g^{162} & g^{166} & g^{85} & g^{212} & 233 \\
g^{81} & g^{68} & 0 & g^{136} & g^{69} & g^{77} & g^{170} & 165 \\
g^{83} & g^{162} & g^{136} & 0 & g^{17} & g^{138} & g^{154} & 68 \\
g^{170} & g^{166} & g^{69} & g^{17} & 0 & g^{34} & g^{235} & 250 \\
g^{106} & g^{85} & g^{77} & g^{138} & g^{34} & 0 & g^{151} & 81 \\
g^{84} & g^{212} & g^{170} & g^{154} & g^{235} & g^{151} & 0 & g^{113} \\
1 & g^{233} & g^{165} & g^{68} & g^{250} & g^{81} & g^{113} & 0
\end{pmatrix}
\]

The corresponding APN function of \( H' \) is \( F'(x) = g^{145} x^{192} + g^{173} x^{160} + g^{239} x^{144} + g^{141} x^{136} + g^{197} x^{132} + g^{35} x^{130} + g^{92} x^{129} + g^7 x^{96} + g^{176} x^{80} + g^{99} x^{72} + g^{135} x^{68} + g^{182} x^{66} + g^{124} x^{65} + g^{117} x^{48} + g^9 x^{40} + g^{108} x^{36} + g^{160} x^{34} + g^{187} x^{33} + g^3 x^{24} + g^{27} x^{20} + g^{156} x^{18} + g^{215} x^{17} + g^{99} x^{12} + g^{188} x^{10} + g^9 x^9 + g^{137} x^6 + g^{225} x^5 + g^{206} x^3 \), which is CCZ-inequivalent to \( x^3 \). So we have got a new APN functions over \( F_2^n \).
5.3 Experimental results

We have implemented the algorithm in this paper. In this subsection we will report experiment results using our algorithm.

(i) Dillon [12] listed 18 classes of CCZ-inequivalent APN functions over $\mathbb{F}_{2^7}$. Edel [16] found a new class of APN function and this list expanded to 19 classes. With our method, firstly we can construct a $7 \times 7$ QAM $H$ from $x^3$, then reassign the values $H[3, 6]$, $H[3, 7]$, $H[4, 5]$, $H[4, 6]$, $H[4, 7]$, $H[5, 6]$, $H[5, 7]$ and $H[6, 7]$ (during this process, we must keep $H$ symmetric). Using this idea we can get more than 470 classes of CCZ-inequivalent quadratic APN functions, these functions are all CCZ-inequivalent to the known ones. Similar method can be used on $\mathbb{F}_{2^6}$. According to Edel’s results [14], there is only 13 classes of CCZ-inequivalent quadratic APN functions. Our program shows that it is only need to change 8 $(2 \times 4)$ elements of a QAM and get all the 13 classes of CCZ-inequivalent quadratic APN functions. This method is an early version of our program, which works efficiently on $\mathbb{F}_{2^n}$ for $n \leq 7$, but when $n \geq 8$, it becomes very slow.

(ii) For now, the set model introduced in Problem 1 is the most efficient method for finding new QAMs. It should be noted that we must change the last two columns (and rows) of a known QAM to get new QAMs when $n \geq 8$. Fortunately, Algorithm 1 can be implemented in parallel, and we are running our program in many computers now. Up to now, we have found more than 1000 classes of CCZ-inequivalent quadratic APN functions on $\mathbb{F}_{2^8}$, and they are all CCZ-inequivalent to the 23 classes of known ones introduced by Dillon [12] and Edel [16]. We have checked all these new APN functions with the method introduced in [4], none of them is CCZ-equivalent to a permutation.

Checking the equivalence of APN functions is equal to check the equivalence of some corresponding codes [3]. Our program about checking CCZ-equivalence is based on this result.

6 Conclusion

We find a one to one correspondence between restricted quadratic APN functions and QAMs. Based on this correspondence, we propose the notion of proper matrix. The most important part of our algorithm is how to keep the matrix proper during the construction process. Algorithm 1 is the core part of our searching program. As a matter of fact, we have designed some slower algorithms before Algorithm 1, and the APN functions on $\mathbb{F}_{2^7}$ are all found by these algorithms. There is much redundant calculation in these algorithms, so we omit them. We find many properties about QAMs and proper matrices, and we have listed them in section 4. The process of finding these properties is essentially the process of speeding our algorithms. Thus, the readers must have a profound understanding of these properties in order to understand Algorithm 1.

Up to now, we have constructed more than 470 and 1000 classes of new CCZ-inequivalent quadratic APN functions on $\mathbb{F}_{2^7}$ and $\mathbb{F}_{2^8}$ respectively (See the appendices of [22]). We think our lists are not complete, especially the list on $\mathbb{F}_{2^8}$, which is far from complete. So we will add some new quadratic APN functions in the lists in the future. Certainly we will also check whether the new APN functions are CCZ-equivalent to some permutations. Our idea is just a beginning. Much related
work can be done in the future, such as, finding some QAMs whose corresponding functions are APN on $\mathbb{F}_{2^n}$ for infinite many $n$, generalizing the matrix approach to construct PN functions, and finding better methods to construct QAMs, etc.

References

On Quadratic Almost Perfect Nonlinear Functions and Their Related Algebraic Object

Guobiao Weng · Yin Tan · Guang Gong

Abstract It is well known that almost perfect nonlinear (APN) functions achieve the lowest possible differential uniformity for functions defined on fields with even characteristic, and hence, from this point of view, they are the most ideal choices for S-boxes in block and stream ciphers. They are also interesting as the link to many other areas, for instance topics in coding theory and combinatorics. In this paper, we present a characterization of quadratic APN functions by a certain kind of algebraic object, which is called an APN algebra. By this characterization and with the help of a computer, we discovered 285 new (up to CCZ equivalence) quadratic APN functions on $\mathbb{F}_{2^7}$, which is a remarkable contrast to the currently known 17 such functions. Furthermore, 10 new quadratic APN functions on $\mathbb{F}_{2^8}$ are found. We propose some problems and conjectures based on the computational results.

Keywords Almost perfect nonlinear · Quadratic function · Substitution box · Algebra

1 Introduction

In the modern design of block and stream ciphers, functions defined on finite fields are chosen as Substitution boxes (or S-box in short) to bring the confusion to the cipher. The S-box needs to be designed carefully to avoid many attacks on the cipher. For instance, the S-boxes are required to be with low differential uniformity (defined below) to prevent the differential cryptanalysis proposed by Biham and Shamir [1].

It is well-known that almost perfect nonlinear (or APN in short, see definition in Section II) functions achieve the lowest possible differential uniformity for functions defined on fields with even characteristic, and therefore, from this point of view, they are the most ideal choices for S-boxes. Such functions are not only interesting in the cryptography, they are demonstrated to be linked with many other topics in the theory of sequences [16], difference sets, bent functions [20] and finite geometry [15].

APN functions were firstly introduced by Nyberg in [18], and several such functions were constructed in her paper. Since then, many power APN functions are discovered by various researchers, see Table 1 for all known families. It is conjectured that the list of power APN functions is complete (up to CCZ-equivalence, see the definition in Section II).

G. Weng
School of Mathematical Sciences, Dalian University of Technology, Liaoning 116024, China.
E-mail: gbweng@dlut.edu.cn

Y. Tan and G. Gong
Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada.
E-mail: {y24tan, ggong}@uwaterloo.ca
Table 1 Known power APN functions on $\mathbb{F}_{2^n}$

<table>
<thead>
<tr>
<th>Exponent $d$</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^i + 1$</td>
<td>$\gcd(i, n) = 1$</td>
</tr>
<tr>
<td>$2^{2i} - 2^i + 1$</td>
<td>$\gcd(i, n) = 1$</td>
</tr>
<tr>
<td>$2^t + 3$</td>
<td>$n = 2t + 1$</td>
</tr>
<tr>
<td>$2^t + 2^{(3t+1)/2} - 1$, $t$ even</td>
<td>$n = 2t + 1$</td>
</tr>
<tr>
<td>$2^t - 2$, $t$ odd</td>
<td>$n = 2t + 1$</td>
</tr>
<tr>
<td>$2^{n} - 2$</td>
<td>$n = 5t$</td>
</tr>
</tbody>
</table>

Besides the preceding power APN functions, in [12], two sporadic binomial APN functions defined on $\mathbb{F}_{2^{10}}$: $x^3 + \omega x^{36}$, where $\omega$ has order 3 or 93; $x^3 + \omega x^{36}$, where $\omega$ has order 273 or 585 are discovered. These two examples are verified to be CCZ-inequivalent to any power functions in Table 1. They were soon generalized into infinite families in [4]. In 2006, Dillon gave more sporadic APN polynomials on $\mathbb{F}_{2^8}$ which are not CCZ-equivalent to any power functions in his talk [10]. These sporadic examples became a source of obtaining new infinite families of APN functions. Many new infinite families are successfully discovered in the sequel, see [2, 3, 6, 7] and the references therein. We should note that all infinite families of APN functions constructed since 2005 are quadratic ones (see definition in Section II).

Besides the method of generalizing sporadic APN functions into APN polynomials, in [5], the switching method is applied to construct new APN functions and the APN function $x^3 + \text{Tr}(x^9)$ was found. This function is beautiful in the sense that it is obtained via changing one component function of a known APN function $x^3$. The switching method was further explored in [13] and more sporadic APN functions were discovered. More precisely, they discovered one new APN function on $\mathbb{F}_{2^5}$, one new on $\mathbb{F}_{2^7}$ and eleven on $\mathbb{F}_{2^8}$. At this stage, due to the discovery of many quadratic APN functions, the following question was proposed in [13]:

**Problem 1** Does the number of CCZ inequivalent APN functions on $\mathbb{F}_{2^n}$ grow exponentially with the increase of $n$?

It is conjectured that the above problem has a positive answer, but the number of known quadratic APN functions so far cannot give a strong evidence of this conjecture. One may refer to Table 2 for the number of APN functions on small fields known so far.

We should note that, comparing to the discovery of many quadratic APN functions, few non-quadratic ones are known (a sporadic example on $\mathbb{F}_{2^6}$ was found in [13]). Moreover, there is little knowledge about the existence of APN permutations on fields with even degrees, i.e. $\mathbb{F}_{2^n}$. Until now, there is only one such function on $\mathbb{F}_{2^6}$ was found by Dillon in [11]. This APN permutation is CCZ-equivalent to the quadratic APN function $x^3 + u^{11}x^6 + ux^9$ on $\mathbb{F}_{2^6}$, where $u$ is a primitive element of $\mathbb{F}_{2^6}$. To discover more APN permutations on $\mathbb{F}_{2^n}$ is called the **BIG APN Problem**. By Dillon’s method, finding more quadratic APN functions may give a hope to obtain APN permutations on $\mathbb{F}_{2^n}$.

In this paper, we present a new characterization of quadratic APN functions. It is shown that such functions are conceptually equivalent to certain algebraic object, which is called an APN algebra which will be introduced in this paper. More precisely, let $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$, where $+$ is the finite field addition and $*$ is a well defined binary operation on $\mathbb{F}_{2^n}$. We call $\mathfrak{A}$ an **APN algebra** if the operation $*$ satisfies the commutative and distributive law, and $x * y = 0$ if and only if $x = y$ or one of $x, y$ is 0 (also defined in Section 3). Now, for a function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, in Theorem 1, we show that if $F$ is a quadratic APN function, $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$ is an APN algebra, where

$$x * y = F(x) + F(y) + F(x + y) + F(0).$$

Conversely, for any APN algebra $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$, a quadratic APN function can be defined through it (see (6)). This characterization enables us to give a unifying treatment of quadratic APN functions in terms of APN algebras.

Furthermore, in Section III, we present a matrix representation of the APN algebra, which can be applied to search for new (up to CCZ equivalence) APN functions on small fields. Surprisingly, by a computer, many
new quadratic APN functions on small fields are discovered. We use the following table to compare the number of APN functions known so far (c.f. [13]) and newly discovered in this paper on $\mathbb{F}_2$, $\mathbb{F}_7$ and $\mathbb{F}_8$.

Several remarks on Table 2 are in the sequel. Firstly, with the help of a computer, we have a proof that only 13 quadratic APN functions on $\mathbb{F}_2$. We should mention that this fact is also known by Yves Edel. Secondly, on $\mathbb{F}_7$, we found 30,000 quadratic APN functions by a personal computer in two days. We randomly choose 5,000 of them to find new quadratic APN functions and test their newness, of which 285 new functions are obtained. We reasonably believe that more new ones may be found in the remaining 25,000 functions. Finally, on $\mathbb{F}_8$, we test 500 quadratic APN functions, and 10 new ones are obtained. All these computations are done by a personal laptop, we expect that more APN functions may be quickly found by a more powerful computer.

The rest of the paper is organized as follows. In Section 2, we give necessary definitions and results used later. The relationship between APN algebras and quadratic APN functions are discussed in Section 3. We also describe the matrix representation of APN algebras and construct an APN algebra there. Section 4 is devoted to explaining the techniques how to search for new quadratic APN functions on small fields using the above characterization. The new quadratic APN functions on $\mathbb{F}_2$, $\mathbb{F}_7$ and other computational results are presented in this Section as well. Based on the computational result, we also propose some open problems and conjectures. Finally, we give some concluding remarks in Section 5.

### Table 2 Number of APN functions on $\mathbb{F}_2^n$ in [13] and in this paper

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\sharp$ of APN in [13]</th>
<th>$\sharp$ of newly found APN in this paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>$\geq 285$</td>
</tr>
<tr>
<td>8</td>
<td>23</td>
<td>10</td>
</tr>
</tbody>
</table>

2 Preliminaries

In this Section, we give some definitions and results which will be used in the following sections.

2.1 Differential and Walsh spectrum

Let $\mathbb{F}_{2^n}$ be a finite field and $\mathbb{F}_{2^n} = \mathbb{F}_{2^n}\setminus\{0\}$. Let $F$ be a function $F$ on $\mathbb{F}_{2^n}$, for any two-tuple $(a,b) \in \mathbb{F}_{2^n}^2 \times \mathbb{F}_{2^n}$, define

$$\delta_F(a,b) = \sharp \{x \in \mathbb{F}_{2^n}: F(x + a) + F(x) = b\},$$

where $\sharp S$ denotes the cardinality for a set $S$. We call

$$\Delta_F := \max_{(a,b) \in \mathbb{F}_{2^n}^2 \times \mathbb{F}_{2^n}} \delta_F(a,b)$$

the differential uniformity of $F$, or call $F$ a differentially $\Delta_F$-uniform function. The multiset $\{\delta_F(a,b) : (a,b) \in \mathbb{F}_{2^n}^2 \times \mathbb{F}_{2^n}\}$ is called the differential spectrum of $F$. In particular, we call $F$ almost perfect nonlinear (APN) if $\Delta_F = 2$.

Another common approach to characterize the nonlinearity of $F$ is as follows. For the function $F$, the Walsh (Fourier) transform $F^W : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \rightarrow \mathbb{C}$ of $F$ is defined by:

$$F^W(a,b) := \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\text{Tr}(aF(x) + bx)},$$

where $\text{Tr}(x) = \sum_{i=0}^{n-1} x^{2^i}$ denotes the absolute trace function. The multiset $W_F := \{F^W(a,b) : a \in \mathbb{F}_{2^n}, b \in \mathbb{F}_{2^n}\}$ is called the Walsh spectrum of $F$. Some researchers call the multiset containing the Walsh spectrum and its negative the extended Walsh spectrum of $F$. Furthermore, it is conjectured that $\max |W_F(a,b)| \geq 2^{(n+1)/2}$ when $n$ is odd, and it is known $\max |W_F(a,b)| \geq 2^{n/2 + 1}$ when $n$ is even. We call $F$ an almost bent (AB) function if
Let instance. For convenience, we review some invariants developed in [13].

For two functions being CCZ-equivalent, but they may imply some properties of APN functions, see Result 2 for and extended Walsh spectrum of APN functions. The same of these invariants are only the necessary condition

$$F^{W}(a, b) \in \{0, \pm 2^{(n+1)/2}\} \text{ for all } a, b \in \mathbb{F}_{2n}^2. \text{ Obviously, AB functions can only exist on } \mathbb{F}_{2n} \text{ with } n \text{ odd. It is well known that any AB function is an APN function ([9]), but not vice versa ([13]). However, any quadratic APN function must be an AB function ([8]).}

Finally, We call the function $F$ quadratic if for all $a \in \mathbb{F}_{2n}^2$, the function

$$L_a(x) \triangleq F(x + a) + F(x) + F(a)$$

is linear.

2.2 EA and CCZ equivalence

Two functions $F$ and $G$ defined on $\mathbb{F}_{2n}^2$ are called extended affine (EA-) equivalent if there exist affine permutations $A_1, A_2: \mathbb{F}_{2n} \rightarrow \mathbb{F}_{2n}$ and an affine function $A$ such that $G = A_1 \circ F \circ A_2 + A$. They are called Charpin-Zinoviev (CCZ) equivalent if their graphs $G_F = \{(x, F(x)) : x \in \mathbb{F}_{2n}^2\}$ and $G_G = \{(x, G(x)) : x \in \mathbb{F}_{2n}^2\}$ are affine equivalent, that is, there exists an affine automorphism $L$ of $\mathbb{F}_{2n} \times \mathbb{F}_{2n}$ such that $L(G_F) = G_G$. It is well known that EA equivalence implies CCZ equivalence, but not vise versa. However, for two quadratic APN functions $F, G$, it is recently shown in [21] that they are CCZ equivalent if and only if they are EA equivalent.

Usually, it is difficult to judge the CCZ equivalence of two APN functions. In [13], a coding theory method to characterize the CCZ equivalence is given. First note that throughout this paper, we always use the identification of the additive group of the vector space $\mathbb{F}_{2n}^2$ with the additive group of the finite field $\mathbb{F}_{2n}$.

More precisely, let $\{\alpha_1, \cdots, \alpha_n\}$ be a basis of $\mathbb{F}_{2n}$ over $\mathbb{F}_2$, for each element $x \in \mathbb{F}_{2n}$, there exist a unique vector $x = (x_1, \cdots, x_n) \in \mathbb{F}_{2n}^n$ such that $x = x_1\alpha_1 + \cdots + x_n\alpha_n$. We use the notation $x$ to denote the corresponding vector in $\mathbb{F}_{2n}^n$ of the element $x \in \mathbb{F}_{2n}^2$.

Let $F$ be an APN function, define the matrix $C_F \in \mathbb{F}_{2n}^{(2n+1)\times 2n}$ as follows:

$$C_F = \begin{bmatrix}
\cdots & 1 & \cdots \\
\cdots & \mathbf{x} & \cdots \\
\cdots & F(\mathbf{x}) & \cdots
\end{bmatrix},$$

where the columns of $C_F$ are ordered with respect to some ordering of the elements of $\mathbb{F}_{2n}$ (in the matrix the elements $\mathbf{x}, F(\mathbf{x})$ are regarded as elements in $\mathbb{F}_{2n}^n$ as explained above). Let $C_F$ be the linear code generated by $C_F$. We have the following result.

Result 1 [13] Let $F, G$ be two APN functions and $C_F, C_G$ be the linear codes generated from them as above. Then $F$ and $G$ are CCZ equivalent if and only if $C_F$ and $C_G$ are equivalent.

There are some invariants of APN functions under CCZ-equivalence. For instance, the differential spectrum and extended Walsh spectrum of APN functions. The same of these invariants are only the necessary condition of two functions being CCZ-equivalent, but they may imply some properties of APN functions, see Result 2 for instance. For convenience, we review some invariants developed in [13].

Using the language of group rings, an APN function $F$ can be denoted by $G_F = \sum_{x \in \mathbb{F}_{2n}^2} (x, F(x))$. It is not hard to see that $F$ is APN if and only if

$$G_F \cdot G_F = 2^n \cdot (0, 0) + 2 \cdot D_F$$

for some $D_F \in \mathbb{F}_{2n} \times \mathbb{F}_{2n} \setminus \{(0, 0)\}$. Denoting by $\text{Dev}(G_F)$ and $\text{Dev}(D_F)$ the two developments (see definition in [13]) of $G_F$ and $D_F$. By [13], if $F$ and $G$ are CCZ equivalent, the designs $\text{Dev}(G_F)$ and $\text{Dev}(D_F)$ are isomorphic. Therefore, the order of automorphism groups and the 2-rank of their incidence matrices (which are denoted by $I$- and $\Delta$- rank respectively) are invariant under CCZ equivalence. Moreover, let $\mathcal{M}(G_F)$ (resp. $\mathcal{M}(D_F)$) be the set of automorphisms of $\mathbb{F}_{2n} \times \mathbb{F}_{2n}$ such that $\sigma(G_F) = G_F \cdot (u, v)$ (resp. $\sigma(D_F) = D_F \cdot (u, v)$) for some $(u, v) \in \mathbb{F}_{2n} \times \mathbb{F}_{2n}$. It is shown in [13] that $\mathcal{M}(G_F)$ and $\mathcal{M}(D_F)$ are groups under the multiplication of $\text{Aut}(\mathbb{F}_{2n} \times \mathbb{F}_{2n})$, which are called multiplier groups and are also invariant under CCZ equivalence.

These parameters are interesting as they imply some properties of the APN function.
Let $F$ be an APN function on $\mathbb{F}_{2^n}$ and $v = \sharp M(G_F)$. Then: (1) $n \cdot (2^n - 1) \mid v$ if $F$ is CCZ equivalent to a power mapping; and (2) $n \mid v$ if $F$ is CCZ equivalent to a polynomial in $\mathbb{F}_2[x]$.

We should mention that one reason for us to introduce the above parameters is that they are easily computed by MAGMA, which helps us study the newly obtained APN functions.

### 3 Quadratic APN functions and APN algebras

In this Section, we will establish a relationship between quadratic APN functions and APN algebras. We first introduce a matrix representation of an APN algebra and then use it to prove the aforementioned relationship. This representation is very useful to search for new quadratic APN functions on small fields, which will be discussed in Section IV. First, we give the definition of the APN algebra.

**Definition 1** Let $\mathbb{F}_{2^n}$ be a finite field and $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$, where $+$ is the finite field addition and $*$ is a well defined binary operation on $\mathbb{F}_{2^n}$. $\mathfrak{A}$ is called APN algebra if the operation $*$ satisfies the commutative and distributive law, and $x * y = 0$ if and only if $x = y$ or one of $x, y$ is 0.

#### 3.1 Matrix representation of APN algebra

Define an $n \times n$ matrix $A$ by

$$A = (a_{ij})_{n \times n}, \quad a_{ij} = \alpha_i \cdot \alpha_j. \quad (3)$$

Note that, by the definition of APN algebra, $a_{ii} = 0$ for all $1 \leq i \leq n$ and $A^T = A$. Now, we may use the matrix $A$ to represent the APN algebra $\mathfrak{A}$ by the following result.

**Proposition 1** Let $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$ be an APN algebra and $A$ be the its corresponding matrix defined in (3). For any elements $x, y \in \mathbb{F}_{2^n}$, we have $x * y = xAY^T$, where $x, y$ are the corresponding vectors of $x, y$ in $\mathbb{F}_{2^n}$.

**Proof** Let $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$. We have

$$x * y = \left(\sum_{i=1}^{n} x_i \alpha_i\right) * \left(\sum_{i=1}^{n} y_i \alpha_i\right)$$

$$= \sum_{i,j=1}^{n} x_i y_j (\alpha_i \cdot \alpha_j)$$

$$= xAY^T.$$

We finish the proof. \hfill \Box

The next result gives a property of the matrix $A$, which is used in Section 4 to search for new APN functions on small fields.

**Proposition 2** Let $\mathfrak{A} = (\mathbb{F}_{2^n}, +, *)$ be an APN algebra and $A$ be the matrix defined in (3). Then for each row (column) of $A$, the $n - 1$ nonzero elements are linearly independent over $\mathbb{F}_2$.

**Proof** Since $A$ is symmetric, we only prove the result is true for each row. Without loss of generality, we prove the $n - 1$ nonzero elements $a_{12}, \cdots, a_{1n}$ in the first row of $A$ are linear independent. Assume there exists $(t_2, \cdots, t_n) \in \mathbb{F}_2^{n-1}$ such that $t_2a_{12} + \cdots + t_na_{1n} = 0$. Substituting $a_{ij} = \alpha_i \cdot \alpha_j$ we have

$$0 = t_2 \alpha_1 \cdot \alpha_2 + \cdots + t_n \alpha_1 \cdot \alpha_n$$

$$= \alpha_1 \cdot (t_2 \alpha_2 + \cdots + t_n \alpha_n).$$

Then, by the definition of APN algebra, we have $t_2 \alpha_2 + \cdots + t_n \alpha_n$ equals 0 or $\alpha_1$. Clearly $t_2 \alpha_2 + \cdots + t_n \alpha_n \neq \alpha_1$ as $\{\alpha_1, \cdots, \alpha_n\}$ is a basis. Similarly, $t_2 \alpha_2 + \cdots + t_n \alpha_n = 0$ if and only if $t_i = 0$ for $2 \leq i \leq n$, which implies that $\alpha_2, \cdots, \alpha_n$ is linear independent. \hfill \Box
Furthermore, for simplicity, we write the matrix $A$ as in the form $A = B + B^T$, where

$$B = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}_{n \times n}. \tag{4}$$

### 3.2 The relationship

Now we are ready to give the main result of this section.

**Theorem 1** Let $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a quadratic APN function. Define the multiplication $*_F : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ by

$$x *_F y = F(x + y) + F(x) + F(y) + F(0). \tag{5}$$

Then $(\mathbb{F}_{2^n}, +, *_F)$ is an APN algebra. Conversely, let $\mathcal{A} = (\mathbb{F}_{2^n}, +, *)$ be an APN algebra. Let the matrices $A, B$ be the ones defined in (3) and (4). Then the function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ defined by

$$F(x) = xBx^T \tag{6}$$

is a quadratic APN function. Moreover, $x *_F y = x * y$ for $x, y \in \mathbb{F}_{2^n}$.

**Proof** First, for the quadratic APN function $F$, we show that $\mathcal{A} = (\mathbb{F}_{2^n}, +, *_F)$ is an APN algebra, where $*_F$ is defined in (5). Clearly, for all $x, y, z \in \mathbb{F}_{2^n}$, we have

$$x *_F y = y *_F x \quad \text{(commutative law)},$$

$$x *_F (y + z) = x *_F y + x *_F z \quad \text{(distributive law)}.$$  

Note that the distributive law is followed from $F(x + a) + F(x) + F(a) + F(0)$ is linear for any nonzero $a$ as $F$ is quadratic. It is also clear that $x *_F y = 0$ if and only if $x = y$ or one of $x, y$ is 0. Indeed, assume that $x *_F y = F(x + y) + F(x) + F(y) + F(0) = 0$ and $y \neq 0$, we have $F(x + y) + F(x) = F(y) + F(0)$. It then follows from $F$ is an APN function that $x = 0$ or $x = y$.

Conversely, for the APN algebra $\mathcal{A} = (\mathbb{F}_{2^n}, +, *)$, we need to show the function $F$ defined in (6) is a quadratic APN function. Obviously, $F$ is quadratic. For any nonzero $a \in \mathbb{F}_{2^n}$, we need to demonstrate the equation

$$\Delta_a(x) = F(x + a) + F(x) + F(a) = 0 \tag{7}$$

has at most two solutions. Substituting $F$ defined in (6) into (7) we get

$$0 = \Delta_a(x) = xBa^T + aBx^T = xBa^T + (aBx^T)^T = x(B + B^T)a^T = xAa^T = x * a.$$

Since $\mathcal{A}$ is an APN algebra, then from above we have $x = 0$ or $a$, which follows that $F$ is an APN function. Finally, it is easy to verify that $x * y = x *_F y$ for all $x, y \in \mathbb{F}_{2^n}$. We finish the proof. \qed

**Remark 1** Expanding the quadratic APN function in (6), we may write $F$ as

$$F(x) = \sum_{i,j=1}^{n} x_i x_j (\alpha_i * \alpha_j), \quad x = (x_1, \cdots, x_n).$$
Two APN algebras $\mathfrak{A}_1 = (F_{2^n}, +, \ast_1)$ and $\mathfrak{A}_2 = (F_{2^n}, +, \ast_2)$ are said to be isomorphic if there exist two linear permutations $L_1, L_2$ such that

$$L_1(x) \ast_1 L_1(y) = L_2(x \ast_2 y)$$

for all $x, y \in F_{2^n}$. The following result shows that two quadratic APN functions are EA equivalent (or equivalently, CCZ equivalent by [21]) if and only if their corresponding APN algebras are isomorphic.

**Theorem 2** Let $F_1, F_2$ be two quadratic APN functions on $F_{2^n}$. If $F_1$ and $F_2$ are EA equivalent, their corresponding APN algebras $\mathfrak{A}_1 = (F_{2^n}, +, \ast_{F_1})$ and $\mathfrak{A}_2 = (F_{2^n}, +, \ast_{F_2})$ are isomorphic. Conversely, if two APN algebras $\mathfrak{A}_1 = (F_{2^n}, +, \ast_1)$ and $\mathfrak{A}_2 = (F_{2^n}, +, \ast_2)$ are isomorphic, their corresponding APN functions $F_1, F_2$ defined in (6) are EA equivalent.

**Proof** Suppose that $F_1$ and $F_2$ are EA-equivalent, then there exist affine permutations $A_1, A_2$ and an affine function $A_3$ such that

$$F_1 \circ A_1 + A_3 = A_2 \circ F_2$$

(8)

Let $A_1(x) = L_1(x) + c_1$ and $A_2(x) = L_2(x) + c_2$, where $L_1, L_2$ are linear permutations. Substituting $A_1, A_2$ into (8), we may get

$$F_1(A_1(x)) + F_1(A_1(y)) + F_1(A_1(x + y)) + F_1(A_1(0)) = L_2(F_2(x) + F_2(y) + F_2(x + y) + F_2(0)).$$

Since $F_1$ is quadratic, the above equation may be simplified as

$$F_1(L_1(x)) + F_1(L_1(y)) + F_1(L_1(x + y)) + F_1(L_1(0)) = L_2(F_2(x) + F_2(y) + F_2(x + y) + F_2(0)),$$

which follows that $L_1(x) \ast_{F_1} L_1(y) = L_2(x \ast_{F_2} y)$ and hence $\mathfrak{A}_1, \mathfrak{A}_2$ are isomorphic.

Conversely, let $\mathfrak{A}_1, \mathfrak{A}_2$ be two isomorphic APN algebras and $F_1, F_2$ be their corresponding quadratic APN functions. To show that $F_1, F_2$ are EA equivalent, we need to demonstrate that there exist affine permutations $A_1, A_2$ such that $F_1 \circ A_1 + A_2 \circ F_2$ is affine. Since $\mathfrak{A}_1, \mathfrak{A}_2$ are isomorphic, there exist linear permutations $L_1, L_2$ such that

$$L_1(x) \ast_1 L_1(y) = L_2(x \ast_2 y).$$

By Theorem 1, we have

$$L_1(x) \ast_{F_1} L_1(y) = L_2(x \ast_{F_2} y).$$

Expanding the above equation we may see that $F_1 \circ L_1 + L_2 \circ F_2$ is affine. We finish the proof. \hfill \Box

3.3 A construction of APN algebra $(F_{2^{2k}}, +, \ast)$

In the following, we give an example of APN algebra.

**Theorem 3** Let $F_{2^n}$ be a finite field with $n = 2k$ and write $F_{2^n} = F_{2^{2k}} \times F_{2^k}$. Let $s$ be an integer with $\gcd(s, n) = 1$. For any $x = (a, b), y = (c, d) \in F_{2^n}$, define $x \ast y$ as

$$x \ast y = \left(ad + bc, t_0(ac^2 + a^2c) + t_1(a^2d + c^2b) + t_2(ad^2 + cb^2) + t_3(b^2d + bd^2)\right),$$

(9)

where the polynomial

$$t_0x^{2s+1} + t_1x^{2s} + t_2x + t_3 \in F_{2^n}[x]$$

(10)

has no zeros over $F_{2^n}$. Then $\mathfrak{A} = (F_{2^n}, +, \ast)$ is an APN algebra.
The commutative and distributive law can be verified easily for $\mathbb{F}$. It is also clear that $x \cdot y = 0$ if $x = y$ or one of $x, y$ is zero. Now, assume that $x \cdot y = 0$, we need to show that either $x = y$ or one of $x, y$ is 0. W.l.o.g. suppose $x \neq 0$, then

$$0 = ad + bc,$$

$$0 = t_0(acx^2 + ax^2c) + t_1(a^2x^2 + c^2x^2) + t_2(adx^2 + cx^2b) + t_3(b^2x^2 + dbx^2).$$

By (11), the determinant

$$0 = \begin{vmatrix} a & c \\ b & d \end{vmatrix},$$

which follows that $c = ta, d = tb$ for some $t \in \mathbb{F}_{2k}$. Substituting them in (12) we have

$$(t + t^2)(t_0a^{2x^2+1} + t_1a^2b^{2x^2} + t_2b^2d + d) = 0.$$  

Dividing $b^{2x+1}$ across the above equation we obtain

$$(t + t^2)\left(t_0 \left(\frac{a}{b}\right)^{2x+1} + t_1 \left(\frac{a}{b}\right)^2 + t_2 \left(\frac{a}{b}\right) + t_3\right) = 0.$$  

By the assumption that the polynomial $t_0a^{2x^2+1} + t_1a^2b^{2x^2} + t_2b^{2x^2+1} + t_3b^2 + d$ has no zero over $\mathbb{F}_{2k}$, we can only have $t + t^2 = 0$, which follows that $t \in \mathbb{F}_{2k}$. Now, from $c = ta, d = tb$, we have: $x = y$ when $t = 1$ and $y = 0$ when $t = 0$. We finish the proof.

Remark 2 The existence of the polynomial of the form (10) with no zeros over $\mathbb{F}_{2k}$ can be seen as follows. Firstly, it is clear that there exist $t_0, t_1, t_2$ such that the polynomial $t_0a^{2x^2+1} + t_1a^2b^{2x^2} + t_2b^{2x^2+1} + t_3b^2 + d$ has no zero over $\mathbb{F}_{2k}$, as otherwise it follows that $t_0a^{2x^2+1} + t_1a^2b^{2x^2} + t_2b^{2x^2+1} + t_3b^2 + d$ is a permutation, which is a contradiction.

Corollary 1 Let $F : \mathbb{F}_{2k} \to \mathbb{F}_{2k}$ be the function defined by

$$F(x) = \sum_{i,j=1}^{n} x_i \cdot x_j (\alpha_i \ast \alpha_j), \ x = (x_1, \cdots, x_{2k}),$$

where the multiplication is defined in (9). Then $F$ is a quadratic APN function.

By MAGMA, for small values, we verified the newness of the APN function in Corollary 1. It is found that, when $k$ is even, $F$ is equivalent to the monomial one in [6, Theorem 1]; and when $k$ is even, $F$ is equivalent to the hexanomial one in [3, Theorem 3]. The following Table 3 lists the computational results of the APN function $F$. Recall that the notations of $\Gamma$-rank, $\Delta$-rank, $Dev(G_F)$, $Dev(D_F)$ and $\mathcal{M}(G_F)$ are defined in Section 2.2. The symbol “-” means that our computer cannot calculate that parameter.

In general, we cannot prove the CCZ-equivalence of the APN functions from Corollary 1 to the ones in [6, Theorem 1] and in [3, Theorem 3]; we left this as an open problem.

Problem 2 To show that, when $k$ is odd, the APN functions in Corollary 1 is CCZ-equivalent to the one in [3, Theorem 3]; when $k$ is even, they are CCZ-equivalent to the one in [6, Theorem 1].

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\Gamma$-rank</th>
<th>$\Delta$-rank</th>
<th>$\mathcal{M}(G_F)$</th>
<th>$\mathcal{M}(G_F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1140</td>
<td>94</td>
<td>$2^6 \cdot 5^2$</td>
<td>$2^6 \cdot 5^2$</td>
</tr>
<tr>
<td>8</td>
<td>13200</td>
<td>414</td>
<td>$2^8 \cdot 7^2$</td>
<td>$2^8 \cdot 7^2$</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
<td>$2^{10} \cdot 3^2 \cdot 7$</td>
<td>$2^{10} \cdot 3^2 \cdot 7$</td>
</tr>
</tbody>
</table>
4 New quadratic APN functions

In this Section, we will present the new quadratic APN functions discovered by a computer on $F_2^7$ and $F_2^8$. Some interesting properties of these newly found functions are discussed as well. Firstly, we explain the techniques to discover these new functions.

By Theorem 1, finding a quadratic APN function $F$ is equivalent to finding its corresponding APN algebra $\mathfrak{A}$. Furthermore, by Proposition 1, an APN algebra $\mathfrak{A}$ can be represented as a matrix of the form

$$A = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{12} & 0 & a_{23} & \cdots & a_{2n} \\ a_{13} & a_{23} & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & \cdots & 0 \end{pmatrix}_{n \times n} \quad (13)$$

where $a_{ij} = \alpha_i \ast \alpha_j$ for $1 \leq i, j \leq n$.

Let $\{\alpha_1, \cdots, \alpha_n\}$ be a basis of $F_2^n$ over $F_2$. For each element $x \in F_2^n$, its corresponding vector $x = (x_1, \cdots, x_n) \in F_2^n$ can be uniquely represented as an integer $x = 2^{x_1} + 2^{x_2} + \cdots + 2^{x_n} - 1$. Conversely, each integer in the range $[0, 2^n - 1]$ corresponds to a vector in $F_2^n$ by writing it as the 2-adic form. Clearly, the basis $\{\alpha_1, \cdots, \alpha_n\}$ of $F_2^n$ over $F_2$ corresponds to the integers $1, 2, \cdots, 2^n - 1$. By Proposition 2, the nonzero elements of each row of $A$ are linearly independent over $F_2$. Therefore, we may fix the first row of $A$ to be $\{2, 0, 0, 0, 0, 0, 0\}$ for $2 \leq i \leq n$. By choosing certain second row, and then let the computer do the search. As a result, many new quadratic APN functions on $F_2^7$ and $F_2^8$ are found, which will be presented in Tables 4 and 5 below. To simplify the expression of APN functions, we use a sequence instead of a matrix to represent it. An example below is used to illustrate the expression.

**Example 1** Let $F$ be a quadratic APN function defined on $F_2^7$ and $\mathfrak{A}$ be its corresponding APN algebra. Assume the matrix $A$ of $\mathfrak{A}$ is

$$A = \begin{pmatrix} 0 & 2 & 4 & 8 & 16 & 32 & 64 \\ 2 & 0 & 48 & 35 & 76 & 51 & 69 \\ 4 & 48 & 0 & 1 & 2 & 15 & 104 \\ 8 & 35 & 1 & 0 & 71 & 126 & 13 \\ 16 & 76 & 2 & 71 & 0 & 62 & 28 \\ 32 & 51 & 15 & 126 & 62 & 0 & 70 \\ 64 & 69 & 104 & 13 & 28 & 70 & 0 \end{pmatrix}.$$  

We represent $A$ by the sequence of its nonzero elements of the upper-triangle matrix from left to right and top to bottom, i.e. $[2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 15, 104, 71, 126, 13, 62, 28, 70]$.

4.1 New quadratic APN functions on $F_2^7$

Firstly, we give newly discovered quadratic APN functions on $F_2^7$. By fixing the first row of the matrix $A$ in (13) to be $\{0, 2, 4, 8, 16, 32, 64\}$, and choosing the second row to be $\{2, 0, 48, 35, 76, 51, 69\}$, we found 30,000 quadratic APN functions in two days. After spending seven days on verifying their newness (up to CCZ equivalence) of 5,000 of them, we obtain 285 new ones, which is a remarkable contrast to the currently known 17 such functions! We believe that the remaining 25,000 functions may still yield more new ones.

Instead of listing all 285 new APN functions, we give 10 of them in Table 4 to illustrate their properties, please refer to [19] for a complete list of the computational results. The $\Gamma$, $\Delta$-ranks, the order of the multiplier groups $\mathcal{M}(G_F)$ are computed as well.

Based on the computational results, we discuss some properties of newly obtained APN functions as follows.
### Table 4 New APN functions on $\mathbb{F}_2^7$

<table>
<thead>
<tr>
<th>No.</th>
<th>APN Function</th>
<th>$\Gamma$-rank</th>
<th>$\Delta$-rank</th>
<th>$\mathcal{M}(G_F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 15, 104, 71, 126, 13, 62, 28, 70 }$</td>
<td>4048</td>
<td>212</td>
<td>2^1</td>
</tr>
<tr>
<td>2</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 15, 42, 25, 70, 82, 4, 7, 53 }$</td>
<td>4048</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 44, 115, 124, 39, 71, 120, 22, 110, 12 }$</td>
<td>4048</td>
<td>212</td>
<td>2^1</td>
</tr>
<tr>
<td>4</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 44, 115, 39, 71, 120, 22, 110, 12 }$</td>
<td>4048</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 15, 24, 12, 113, 106, 62 }$</td>
<td>4046</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 15, 24, 12, 113, 106, 62 }$</td>
<td>4046</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 29, 60, 97, 82, 37, 100, 67, 66 }$</td>
<td>4046</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 26, 88, 62, 106, 18, 122, 45, 8 }$</td>
<td>4046</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 12, 71, 48, 86, 125, 46, 66, 37 }$</td>
<td>4050</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>${ 2, 4, 8, 16, 32, 64, 48, 35, 76, 51, 69, 1, 2, 12, 71, 48, 86, 125, 46, 66, 37 }$</td>
<td>4046</td>
<td>212</td>
<td>2</td>
</tr>
</tbody>
</table>

#### 4.1.1 AB function are EA equivalent to a PP

It is conjectured in [8] that any AB function is EA equivalent to a permutation. A counter example (with algebraic degree 3) to this conjecture was given in [2, Theorem 1]. Since quadratic APN functions on fields with odd degrees are AB functions, for each newly discovered APN function $F$ on $\mathbb{F}_2^7$, by a computer, we may find a linearized polynomial $L \in \mathbb{F}_2^7[x]$ such that $F + L$ is a permutation. Therefore, we revise the conjecture in [8] as follows.

**Conjecture 1** Each quadratic AB function is EA-equivalent to a permutation.

#### 4.1.2 Classes of switching neighbors

For an APN function $F$ on $\mathbb{F}_2^n$, we call the APN functions of the form $F(x) + uf(x)$ the *switching neighbors* of $F$ in the narrow sense, where $u \in \mathbb{F}_2^n$ and $f$ is a Boolean function. The class of the switching neighbors of $F$ refers to all CCZ-inequivalent APN functions amongst them. In [13], the authors observed that, on $\mathbb{F}_2^7$, there only have small switching classes (with size 1, 2 or 3), while, on $\mathbb{F}_2^8$, there is one large switching class (with size 17).

For the 285 newly found APN functions on $\mathbb{F}_2^7$, we compute their switching neighbors and further confirm the aforementioned observation, i.e. there is no large switching classes (the largest class has 3 inequivalent APN functions).

#### 4.1.3 Order of $\mathcal{M}(G_F)$

One may see in Table 4 that the order of multiplier group of $G_F$ are all $2^7$. Actually this is true for all found 30,000 APN functions. By Result 2, these newly found APN functions are not CCZ equivalent to any power mappings and they are not in $\mathbb{F}_2^n$. This shows the limit of searching APN functions according to the number of terms of their polynomial form as from the computational results most APN functions have many terms.

#### 4.2 New quadratic APN functions on $\mathbb{F}_2^8$

The new 10 APN functions are presented in Table 5 below. These functions are obtained by choosing the first row of $A$ in (13) to be $\{ 0, 1, 6, 204, 20, 142, 78, 85 \}$ and the second row to be $\{ 1, 0, 204, 202, 154, 20, 85, 29 \}$. We should mention that, on $\mathbb{F}_2^n$, a computer takes much more time to search for new APN functions. However, by choosing the second row more carefully, it is possible to find more quadratic APN functions. Comparing to number of APN functions found in $\mathbb{F}_2^7$, we conjecture there exist thousands of APN functions on $\mathbb{F}_2^8$. Finally, the 10 newly found APN functions on $\mathbb{F}_2^8$ cannot yield APN permutation using Dillon’s method described in [11].
Table 5  New APN functions on $F_{2^8}$

<table>
<thead>
<tr>
<th>No.</th>
<th>APN Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 9, 123, 37, 37, 94, 203]$</td>
</tr>
<tr>
<td>2</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 34, 39, 209, 209, 246, 175]$</td>
</tr>
<tr>
<td>3</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 61, 140, 148, 148, 24, 198]$</td>
</tr>
<tr>
<td>4</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 69, 146, 3, 3, 145, 227]$</td>
</tr>
<tr>
<td>5</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 113, 168, 234, 234, 66, 177]$</td>
</tr>
<tr>
<td>6</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 115, 82, 127, 127, 45, 5]$</td>
</tr>
<tr>
<td>7</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 156, 115, 112, 112, 3, 193]$</td>
</tr>
<tr>
<td>8</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 61, 30, 46, 157, 160, 252, 221, 182, 182, 107, 26]$</td>
</tr>
<tr>
<td>9</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 73, 30, 46, 233, 160, 98, 33, 114, 114, 83, 4]$</td>
</tr>
<tr>
<td>10</td>
<td>$[1, 6, 204, 20, 142, 72, 85, 204, 202, 154, 20, 85, 29, 8, 48, 30, 160, 105, 30, 46, 201, 160, 198, 223, 242, 242, 66, 5]$</td>
</tr>
</tbody>
</table>

Table 6  Invariants of the New APN functions on $F_{2^8}$

<table>
<thead>
<tr>
<th>No.</th>
<th>$Γ$-rank</th>
<th>$Δ$-rank</th>
<th>$♯\mathcal{M}(Γ_F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14040</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>2</td>
<td>14044</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>3</td>
<td>14052</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>4</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>5</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>6</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>7</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>8</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>9</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
<tr>
<td>10</td>
<td>14054</td>
<td>438</td>
<td>$2^3 \cdot 3$</td>
</tr>
</tbody>
</table>

5 Concluding remarks

It is well known that, for functions defined on $F_{p^n}$, where $p$ is an odd prime, the lowest possible differential uniformity is 1 and the functions achieving this value are called perfect nonlinear (PN). The quadratic perfect nonlinear functions are proven conceptually equivalent to commutative semifields. The 3-tuple $A = (F_{p^n}, +, *)$ is a commutative semifield if $*$ satisfies commutative and distributive law, and for $x, y \in F_{p^n}$, $x * y = 0$ if and only if $x = 0$ or $y = 0$. Obviously, the difference between semifield and APN algebra defined in this paper is the condition of $x * y = 0$. The characterization of quadratic APN functions using APN algebra is similar to the treatment of quadratic PN functions.

In this paper, on small fields, we use the relationship between quadratic APN functions and APN algebras, and with the help of a computer, to find 285 new quadratic APN functions on $F_{2^7}$ and 10 new ones on $F_{2^8}$, which is a remarkable contrast to the number of currently known such functions. By the searching techniques developed in Section 4, it is very hopeful to discover much more APN functions on small fields.

After finishing this work, we were informed by a reviewer that a similar work has been done by Y. Yu, M. Wang and Y. Li. We verified the CCZ equivalence of our APN functions on $F_{2^7}$ with theirs, and find that 99 ones are equivalent to theirs. The complete test of our 285 functions on $F_{2^7}$ will take much more time.

References

13. Y. Edel and A. Pott, A new almost perfect nonlinear function which is not quadratic, Advances in Mathematical Communications 3(1), 59–81, (2009).
19. Y. Tan, New quadratic APN functions on $\mathbb{F}_{2^7}$ and $\mathbb{F}_{2^8}$, “https://ece.uwaterloo.ca/~y24tan/”.
Covering Sets for Limited-Magnitude Errors

Torleiv Kløve and Moshe Schwartz

1 T. Kløve, Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Torleiv.Klove@ii.uib.no
2 M. Schwartz, Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva 84105 Israel. schwartz@ee.bgu.ac.il

Abstract. The concept of a covering set for the limited-magnitude error channel is introduced. A number of covering-set constructions, as well as some bounds, are given. In particular, optimal constructions are given for some cases involving small-magnitude errors.

1 Introduction

For integers \(a, b\), where \(a \leq b\), we let
\[
[a, b] = \{a, a + 1, a + 2, \ldots, b\}, \quad [a, b]^* = \{a, a + 1, a + 2, \ldots, b\} \setminus \{0\}.
\]

Throughout this paper, let \(\mu, \lambda\) be integers such that \(0 \leq \mu \leq \lambda\), and let \(q\) be a positive integer. In the \((\lambda, \mu; q)\) limited-magnitude error channel an element \(a \in \mathbb{Z}_q\) can be changed into any element in the set \(\{(a + e) \mod q \mid e \in [-\mu, \lambda]\}\).

For any \(S \subseteq \mathbb{Z}_q\) we define \(MS = \{xs \in \mathbb{Z}_q \mid x \in M, s \in S\}\), where multiplication is done modulo \(q\). If \(|MS| = (\mu + \lambda)|S|\), then \(S\) is packing set. A packing set \(S\) where \(MS \subseteq \mathbb{Z}_q \setminus \{0\}\) is a \(B[-\mu, \lambda](q)\) set in the terminology of [8].

If \(s = (s_1, s_2, \ldots, s_n)\), where \(\{s_1, s_2, \ldots, s_n\}\) is a \(B[-\mu, \lambda](q)\) set, then
\[
\{x \in \mathbb{Z}_q^n \mid x \cdot s \equiv 0 \pmod{q}\}
\]
is a code that can correct a single limited-magnitude error from the set \([-\mu, \lambda]\).

Such codes have been studied in, e.g., [1]-[6], and [8].

Similar to packing sets, we can consider covering sets, where a set \(S\) is called a \((\lambda, \mu; q)\) covering set if \(|MS| = q\). Thus, covering sets are to packing sets as covering codes are to error-correcting codes. Instead of trying to pack many disjoint translates \(Ms, s \in S\), into \(\mathbb{Z}_q\), in the covering set scenario we are interested in having the union of \(Ms, s \in S\), cover \(\mathbb{Z}_q\) entirely with \(S\) being as small as possible. Some results where \(\mu = 0\) or \(\mu = \lambda\) are described in [7]. Apart from its independent intellectual merit, solving this problem for \(\mu = 0\) has immediate applications, such as rewriting schemes for non-volatile memories, a simplified version of which we now describe. For a more detailed description the reader is referred to [2] and references therein.

* Supported by The Norwegian Research Council and by ISF grant 134/10.
Consider a set of \( n \) flash memory cells, each capable of storing an integer from \( \mathbb{Z} \). Let \( G \) be some finite abelian group, say \( G = \mathbb{Z}_q \), and some subset \( S = \{s_1, s_2, \ldots, s_n\} \subseteq G \), where we denote \( s = (s_1, s_2, \ldots, s_n) \). Define the decoding mapping \( D : \mathbb{Z}^n \rightarrow \mathbb{Z}_q \) as \( D(x) = x \cdot s \).

If we want to store the value \( v \in \mathbb{Z}_q \) in the \( n \) memory cells, we choose a vector \( x \in \mathbb{Z}^n \) such that \( D(x) = v \), and store the \( i \)-th component of \( x \) in the \( i \)-th cell. If we then want to rewrite this value with \( v' \in \mathbb{Z}_q \), we can choose a different vector \( x' \in \mathbb{Z}^n \) that decodes to \( v' \). Due to the limitations of flash memory, we would like \( x' \) to be in the range \( [x_i - \mu, x_i + \lambda] \), and to leave as many cells as possible unchanged. In the extreme case, we allow only a single cell to change. To be able to allow any value \( v \) to be rewritten with \( v' \) while changing the stored integer in a single cell as above, \( S \) can be taken to be a \((\lambda, \mu; q)\) covering set. This is because we can write \( v' - v = ms_i \) with \( m \in [-\mu, \lambda] \), \( s_i \in S \), and then choose \( x' = x + me_i \), where \( e_i \) is the \( i \)-th standard unit vector. For maximum efficiency, we would like \( S \) to be as small as possible.

We say \( S \) is a \((\lambda, \mu; q)\) perfect covering set if \( MS = \mathbb{Z}_q \), \(|M|(|S| - 1) = q - 1 \), and \( 0 \in S \). In other words, \( S \) is a perfect covering set if, apart from \( 0 \in S \), the products \( ms \) in \( \mathbb{Z}_q \), where \( m \in M \), \( s \in S \), are all distinct, non-zero, and cover all the non-zero elements of \( \mathbb{Z}_q \). These are also called abelian group splittings in the terminology of [7].

Similarly, \( S \) is a perfect packing set if the products \( ms \) in \( \mathbb{Z}_q \), where \( m \in M \), \( s \in S \), are all distinct, non-zero, and cover all the non-zero elements of \( \mathbb{Z}_q \). We note that \( S \) is a perfect covering if and only if \( S \setminus \{0\} \) is a perfect packing set.

The following functions shall be of interest to us:

\[
\nu(q, r) = \nu_{\lambda, \mu}(q, r) = \max_{S \subseteq \mathbb{Z}_q} \{|MS| \mid |S| = r\}, \\
\theta(q) = \theta_{\lambda, \mu}(q) = \max_{r \in \mathbb{N}} \{r \mid \nu(q, r) = (\mu + \lambda)r\}, \\
\omega(q) = \omega_{\lambda, \mu}(q) = \min_{r \in \mathbb{N}} \{r \mid \nu(q, r) = q\}.
\]

Intuitively speaking, \( \nu(q, r) \) expresses the maximum coverage of sets of size \( r \), \( \theta(q) \) is the maximum size of a packing set, and \( \omega(q) \) is the minimum size of a covering set. If \( S \) is a \((\lambda, \mu; q)\) covering set of minimal size \( \omega_{\lambda, \mu}(q) \), we call \( S \) optimal. We first prove some basic monotonicity properties.

**Theorem 1.** Let \( \mu' \) and \( \lambda' \) be integers such that \( -\mu \leq -\mu' \leq 0 \leq \lambda' \leq \lambda \). Then

\[
\nu_{\lambda', \mu'}(q, r) \leq \nu_{\lambda, \mu}(q, r), \quad \theta_{\lambda', \mu'}(q) \geq \theta_{\lambda, \mu}(q), \quad \omega_{\lambda', \mu'}(q) \geq \omega_{\lambda, \mu}(q).
\]

**Proof.** If we denote \( M' = [-\mu', \lambda']^* \) then obviously \( M' \subseteq M \) and therefore \( M'S \subseteq MS \). The claims follow immediately. \( \square \)

We now give a simple lower bound.

**Theorem 2.** We have \( \omega_{\lambda, \mu}(q) \geq \left\lfloor \frac{q}{\lambda + \mu} \right\rfloor \).
Proof. By definition, there exists an optimal covering set $S$. Therefore,

$$q = |MS| \leq (\lambda + \mu)|S| = (\lambda + \mu)\omega_{\lambda,\mu}(q),$$

and the theorem follows. \qed

Example 1. For $\mu = 0$ and $\lambda = 1$ we clearly have $MS = S$ for all sets $S$. Hence, $\nu_{1,0}(q,r) = r$ and $\theta_{1,0}(q) = \omega_{1,0}(q) = q$.

Example 2. Let $\mu = 1$. For $1 \leq r \leq [q/2]$ we have $|M[1,r]| = 2r$. Hence, $\nu(q,r) = 2r$. For $[q/2] + 1 \leq r \leq q$ we have $|M[0,r-1]| = q$. Hence, $\nu(q,r) = q$.

For $\lambda \geq 2$, it seems to be quite complicated to determine $\theta$ and $\omega$ in many cases. For $\lambda = 2$, $\theta_{2,0}(q)$ was determined in [4], $\theta_{2,1}(q)$ in [8], and $\theta_{2,2}(q)$ in [5]. In the next sections we consider $\omega_{2,0}(q)$ and $\omega_{2,1}(q)$. Because of the page limitations, our results on $\omega_{2,2}(q)$ is are not given here.

We first give a general BCH-like upper bound.

Theorem 3. Let $p$ be a prime, and let $g$ be a primitive element in $\mathbb{Z}_p$. If $[-\mu, \lambda]^*$ contains $\delta$ consecutive powers of $g$ then $\omega_{\lambda,\mu}(p) \leq \left\lceil \frac{p-1}{\delta} \right\rceil + 1$.

Proof. One can easily verify that the set

$$S = \{0\} \cup \left\{ g^{\delta i} \mid 0 \leq i \leq \left\lfloor \frac{p-1}{\delta} \right\rfloor - 1 \right\}$$

is indeed a $(\lambda,\mu;g)$ covering set. \qed

Another upper bound is the following.

Theorem 4. If $q$ and $r$ are odd, then $\omega_{2,\mu}(qr) \leq r(\omega_{2,\mu}(q) - 1) + \omega_{2,\mu}(r)$.

Proof. Let $S$ be an optimal $(2,\mu;q)$ covering set and $D$ an optimal $(2,\mu;r)$ covering set. We remind that $\mu \leq \lambda = 2$. Since $q$ is odd, $ac \equiv 0 \pmod{q}$ for some $a \in [-\mu,2]^*$, only if $c = 0$. Therefore, we must have $0 \in S$. Similarly, $0 \in D$. Let

$$E = \{cq + s \in \mathbb{Z}_q \mid c \in [0,r-1], s \in S \setminus \{0\} \} \cup \{qd \in \mathbb{Z}_q \mid d \in D\}.$$ 

Then $|E| = r(\omega_{2,\mu}(q) - 1) + \omega_{2,\mu}(r)$. We will show that $E$ is a $(2,\mu;q)$ covering set.

First, consider the case $b \in \mathbb{Z}_q$, $b \not\equiv 0 \pmod{q}$. Let $b_1 \equiv b \pmod{q}$, $b_1 \in [1,q-1]$, that is $b = mq + b_1$ for some integer $m$. Furthermore, $b_1 \equiv as \pmod{q}$ for some $a \in [-\mu,2]^*$ and $s \in S \setminus \{0\}$. That is, $as = m_1q + b_1$ for some integer $m_1$. Hence

$$b = mq + (as - m_1q) = (m - m_1)q + as.$$

Since $qr$ is odd, we note that all the elements of $[-\mu,2]$ are invertible in $\mathbb{Z}_q$. Thus,

$$b = (m - m_1)q + as \equiv a^{-1}(m - m_1)q + s \pmod{qr}.$$

This shows that $b \in [-\mu,2]^*E$.

Next, consider the case $b \in \mathbb{Z}_q$, $b \equiv 0 \pmod{q}$. Then $b = qb_2$. There exist $a \in [-\mu,2]^*$ and $d \in D$ such that $b_2 \equiv ad \pmod{r}$. Hence $b = qb_2 \equiv a(qd) \pmod{qr}$, that is $b \in [-\mu,2]^*E$ also in this case. \qed
2 Determination of $\omega_{2,0}(q)$

For $S \subseteq \mathbb{Z}_q$ and $(\lambda, \mu) = (2, 0)$, we have $M = [0, 2]^* = \{1, 2\}$ and

$$MS = \bigcup_{s \in S} \{s, 2s\}.$$  

First, we consider $q = 2m + 1$. For an integer $a \in \mathbb{Z}_{2m+1} \setminus \{0\}$, the corresponding cyclotomic coset is

$$\sigma(a) = \{a2^j \mod (2m + 1) \mid j \geq 0\}.$$  

If $2m + 1$ is a prime, then all the cosets have the same size. We see that a packing set can contain at most $\lfloor |\sigma(a)|/2 \rfloor$ of the elements in $\sigma(a)$, and we can find a packing set with this many elements. Let $\zeta(2m + 1)$ be the number of cyclotomic cosets of odd size. Then we get

$$\theta_{2,0}(2m + 1) = m - \zeta(2m + 1)/2.$$  

This is Theorem 7 in [3], where a more detailed proof is given.

Similarly, a covering set must contain at least $\lceil |\sigma(a)|/2 \rceil$ of the elements in $\sigma(a)$, and we can find a covering set with this many elements. Moreover, a covering set must contain $0$. Hence

$$\omega_{2,0}(2m + 1) = m + 1 + \zeta(2m + 1)/2. \quad (1)$$  

An explicit expression for $\zeta(2m + 1)$ is given as Theorem 2 in [4]. Combining this with (1), we get the following theorem where $\varphi(d)$ is Euler’s function and $\ord_p(2)$ is the multiplicative order of $2$ modulo $p$.

**Theorem 5.** If $2m + 1 = p_1^{t_1}p_2^{t_2} \cdots p_s^{t_s}$ is the prime factorization of $2m + 1$, let $q_o = \prod_{1 \leq i \leq s, p_i \in P_o} p_i^{t_i}$, where $P_o$ is the set of odd primes $p$ such that $\ord_p(2)$ is odd. Then

$$\omega_{2,0}(2m + 1) = m + 1 + \sum_{d \mid q_o, d > 1} \frac{\varphi(d)}{2 \ord_d(2)}.$$  

In particular, a perfect $(2, 0; 2m + 1)$ set exists if and only if none of the primes dividing $2m + 1$ belongs to $P_o$.

**Theorem 6.** For $m \geq 0$ we have $\omega_{2,0}(4m + 2) = 2m + 1$.

**Proof.** By Theorem 2, $\omega_{2,0}(4m+2) \geq 2m+1$. On the other hand, $\{1, 3, \ldots, 4m + 1\}$ is a covering set of size $2m + 1$. \qed

**Theorem 7.** For all $m \geq 1$ we have $\omega_{2,0}(4m) = 2m + \omega_{2,0}(m)$.
Proof. Let $D$ be an optimal $(2,0;m)$ covering set. The set
\[
\{2a + 1 \mid a \in [0, 2m - 1]\} \cup \{4d \mid d \in D\}
\]
is easily seen to be a $(2,0;4m)$ set of size $2m + \omega_{2,1}(m)$. Hence,
\[
\omega_{2,0}(4m) \leq 2m + \omega_{2,0}(m). \tag{2}
\]
On the other hand, let $S$ be an $(2,0;4m)$ covering set. Clearly, $S$ must contain $E = \{2a + 1 \mid a \in [0, 2m - 1]\}$. Let $X$ be the set of even elements in $S$. Let $s \in X$. If $s \equiv 2 \pmod{4}$, then $s \in ME$, where $M = [1, 2]$. Hence we can replace $s$ by $2s$ in $S$ and still have a covering set. Therefore, we may assume that all the elements of $X$ are divisible by 4. Define
\[
D = \{s/4 \mid s \in X\}.
\]
We will show that $D$ is a covering $(2,0;m)$ set. Let $a \in \mathbb{Z}_m$. Then $4a \in \mathbb{Z}_{4m}$. Hence, we have two possibilities:
- $4a \in X$. Then $a \in D$.
- $4a \notin X$. Then $4a = 2 \cdot 4b$, where $4b \in X$, and so $a = 2b$ where $b \in D$.
Hence $MD = \mathbb{Z}_m$. Therefore we get
\[
\omega_{2,0}(4m) = |X| + 2m = |D| + 2m \geq \omega_{2,0}(m) + 2m.
\]
Combined with (2), this proves the theorem. \qed

3 Some results on $\omega_{2,1}(q)$

For $S \subseteq \mathbb{Z}_q$ and $(\lambda, \mu) = (2,1)$, we have $M = [-1,2]^* = \{-1,1,2\}$ and
\[
MS = \bigcup_{s \in S} \{s, -s, 2s\}.
\]

Theorem 8. For all $m \geq 1$ we have $\omega_{2,1}(2m + 1) = m + 1$.

Proof. The set $[0,m]$ is clearly a $(2,1;2m + 1)$ covering set. Hence
\[
\omega_{2,1}(2m + 1) \leq m + 1. \tag{3}
\]

Now, let $S$ be a set of minimal size covering $\mathbb{Z}_{2m+1}$. We note that for $x \in [-1,2]^*$ we have $xs \equiv 0 \pmod{2m + 1}$ if and only if $s = 0$. Hence $0 \in S$. Since $0 \in S$ covers only $0 \in \mathbb{Z}_{2m+1}$ we shall, for the rest of the proof, only consider non-zero elements in $S$ and the covering of non-zero elements in $\mathbb{Z}_{2m+1}$. We partition the elements of $\mathbb{Z}_{2m+1}^*$ into the “positive” and “negative” elements,
\[
P = \{1,2,\ldots,m\} \quad \text{and} \quad N = \{-1,-2,\ldots,-m\}.
\]
We will determine a particular ordering \( s_0 = 0, s_1, s_2, \ldots \) of the elements of \( S \). We use the notation \( S_i = \{s_1, s_2, \ldots, s_i\} \). We shall say \( MS_i \) is of configuration \((j, k)\) if

\[
|P \cap MS_i| = j \quad \text{and} \quad |N \cap MS_i| = k.
\]

We shall further say that a configuration \((j, k)\) is balanced if \( j = k \), almost balanced if \( |j - k| = 1 \), and imbalanced otherwise. We will show by induction that there is an ordering with the following properties:

1. If \( MS_i \) is balanced then:
   (a) \( a \in MS_i \iff -a \in MS_i \).
   (b) \( |MS_i| \leq 2i \).
2. If \( MS_i \) is almost balanced then:
   (a) \( a \in MS_i \iff -a \in MS_i \), except for exactly one element in \( MS_i \).
   (b) \( -2s_i \not\in MS_i \).
   (c) \( |MS_i| \leq 2i + 1 \).
3. \( MS_i \) is never imbalanced.

If \( 2m+1 \) is divisible by 3, then we must have \((2m+1)/3 \in S \) or \(-(2m+1)/3 \in S \) (but not both since \( S \) has minimal size). In this case, we choose this as \( s_1 \). Then

\[
MS_1 = \left\{ \frac{2m+1}{3}, \frac{2m+1}{3}, \frac{2m+1}{3}, \ldots \right\}
\]

which is a balanced set of size 2. Otherwise, \( 2m+1 \) is not divisible by 3, and we choose any non-zero element of \( S \) as \( s_1 \) and we get \( MS_1 = \{s_1, -s_1, 2s_1\} \), an imbalanced set of size 3. Moreover, \( -2s_1 \not\in MS_1 \). Thus, the induction basis is proved.

For the induction step, let us assume the hypothesis holds for \( i \), and we show how to pick \( s_{i+1} \). We consider the following cases:

1. \( MS_i \) is balanced: If we choose as \( s_{i+1} \) an element that is already covered, i.e., \( s_{i+1} \in MS_i \), then by the induction hypothesis \( -s_{i+1} \) is also covered. Now, if \( 2s_{i+1} \) is covered, then again, \( -2s_{i+1} \) is covered and so \( MS_i = MS_{i+1} \) and is balanced. If, on the other hand, \( 2s_{i+1} \) is not covered then so is \( -2s_{i+1} \), but then \( 2s_{i+1} \in MS_{i+1} \) and \( -2s_{i+1} \not\in MS_{i+1} \) and so \( MS_{i+1} \) is almost balanced. If we choose \( s_{i+1} \) that is not covered, then \( -s_{i+1} \) is also not covered. As before, if \( 2s_{i+1} \) is covered, then so is \( -2s_{i+1} \) and \( MS_{i+1} \) is balanced. Otherwise, \( 2s_{i+1} \) is not covered and \( MS_{i+1} \) is almost balanced since \( -2s_{i+1} \not\in MS_{i+1} \).
2. \( MS_i \) is almost balanced: By the induction hypothesis \( -2s_i \not\in MS_i \). We must have \( -2s_i \in \{s, -s, 2s\} \) for some \( s \in S \). We choose \( s_{i+1} \) to be one such \( s \). We therefore have three subcases here to consider:
   (a) \( s_{i+1} = -2s_i \). In that case \( -s_{i+1} \) is already covered. We note that \( 2s_{i+1} \) and \( -2s_{i+1} \) are both covered or both not covered, which results in \( MS_{i+1} \) being balanced or almost balanced (with \( -2s_{i+1} \not\in MS_{i+1} \)) respectively.
   (b) \( s_{i+1} = -2s_i \). This is exactly like the previous case only \( s_{i+1} \) is already covered.
(c) $2s_{i+1} = -2s_i$, that is, $s_{i+1} = -s_i$. In this case both $s_{i+1}$ and $-s_{i+1}$ are already covered, as well as $-2s_{i+1} = 2s_i$ being covered. We now have $2s_{i+1} = -2s_i \in MS_{i+1}$ and $MS_{i+1}$ is balanced.

We note that in all cases we never reach an imbalanced state, and it is a matter of simple bookkeeping to verify the size of $MS_{i+1}$ does not exceed the claim.

Having proved the claims by induction, assume $MS_i = \mathbb{Z}_{2m+1}^*$, i.e., a covering of the non-zero elements of $\mathbb{Z}_{2m+1}^*$. Since $MS_i$ is obviously balanced, by the claims above $i \geq m$. Since we need to add 0 to $S_i$ to get a covering of $\mathbb{Z}_{2m+1}$ we get $\omega_2(2m+1) \geq m + 1$. Combining this with (3), the theorem follows. \(\square\)

**Theorem 9.** For all $m \geq 1$ we have $\omega_2(4m) = m + \omega_2(m)$.

**Proof.** Let $E = \{2a + 1 \mid a \in [0, m-1]\}$. Then

$$ME = \{a \in \mathbb{Z}_{4m} \mid a \not\equiv 0 \pmod{4}\}.$$ Let $D$ be an optimal $(2, 1; m)$ set. Then the set $E \cup \{4d \mid d \in D\}$ is easily seen to be a $(2, 1; 4m)$ set of size $m + \omega_2(m)$. Hence,

$$\omega_2(4m) \leq m + \omega_2(m). \quad (4)$$

On the other hand, let $S$ be an optimal $(2, 1; 4m)$ covering set. Let $S_0$ be the set of even elements in $S$ and $S_1$ be the set of odd elements in $S$. First, we see that for an odd integer $a \in \mathbb{Z}_{4m}$, we must have $a \in S_1$ or $-a \in S_1$. Hence, $S_1$ contains at least $m$ elements. Let $S' = S_0 \cup E$. Then $MS_1 \subseteq ME$ and so $MS' = \mathbb{Z}_{4m}$. Also

$$\omega_2(4m) \leq |S'| = m + |S_0| \leq |S_1| + |S_0| = \omega_2(m),$$

and so $S'$ is an optimal covering set.

Next, if $S_0$ contains an element $s \equiv 2 \pmod{4}$, this covers $s$, $4m - s$, and $s' = (2s \mod{4m})$. The first two are also covered by $E$. Therefore, if we replace $s$ by $s'$, the set is still a covering set for $\mathbb{Z}_{4m}$. Repeating the process with all elements in $S_0$ that are congruent to 2 modulo 4m, we get a set $S'_0$ where all elements are divisible by 4, and such that $E \cup S'_0$ is a covering set, of size $\omega_2(4m)$. Let $D = \{s/4 \mid s \in S'_0\}$. Then it is easy to see that $D$ is a set covering $\mathbb{Z}_m$. Hence, $|S'_0| \geq \omega_2(m)$ and so

$$\omega_2(4m) = |S| = |E| + |S'_0| \geq m + \omega_2(m).$$

Combined with (4), the theorem follows. \(\square\)

The determination of $\omega_2(4m+2)$ seems to be more tricky. We start with a lower bound.

**Theorem 10.** For all $m \geq 1$ we have $\omega_2(4m + 2) \geq 3m/2 + 1.$
Proof. Let $S$ be an optimal $(2, 1; 4m + 2)$ covering set. We first note that the only way to cover $2m + 1 \in \mathbb{Z}_{4m+2}$ is by having $2m + 1 \in S$. Thus, $0$ is also covered since $2(2m + 1) \equiv 0 \pmod{4m + 2}$. We now use an argument similar to that used in the proof of Theorem 9. The odd elements of $\mathbb{Z}_{4m+2}$ can only be covered by odd elements in $S$. Since $s \in S$ covers both $s$ and $-s$, in order to cover the $2m$ remaining odd elements of $\mathbb{Z}_{4m+2}$ we need at least $m$ odd elements in $S$ in addition to our initial choice of $2m + 1 \in S$. Furthermore, this implies that of the $2m$ even non-zero elements of $\mathbb{Z}_{4m+2}$, $m$ are already covered. We are therefore left with $m$ even non-zero elements in $\mathbb{Z}_{4m+2}$ which we still need to cover. Adding an odd element to $S$ can cover at most another single even element in $\mathbb{Z}_{4m+2}$. In contrast, adding an even element $s \in S$ can cover at most two more elements of $\mathbb{Z}_{4m+2}$ since at least one of $s$ and $-s$ is already covered. Thus, we need to add at least $\frac{m}{2}$ more elements to $S$. \hfill $\square$

We turn to prove upper bounds on $\omega_{2,1}(4m + 2)$. Let $v_2$ denote the 2-ary evaluation, that is $n = 2^{v_2(n)} n_1$, where $n_1$ is odd. By an explicit construction, we can find an upper bound on $\omega_{2,1}(4m + 2)$.

**Construction 1.** For $m \geq 0$, let $S = X \cup Y \cup Z$, where

$$
X = \{2a + 1 \mid a \in [0, m]\},
$$

$$
Y = \left\{c \in \left[1, 4 \left\lfloor \frac{m}{3} \right\rfloor + 2 \right] \mid v_2(c) = 1\right\},
$$

$$
Z = \left\{c \in \left[1, 8 \left\lfloor \frac{m}{3} \right\rfloor \right] \mid v_2(c) \text{ is odd and } v_2(c) \geq 3\right\}.
$$

**Proposition 1.** For all $m \geq 0$, $S$ of Construction 1 is a $(2, 1; 4m + 2)$ covering set.

*Proof.* Let $b \in [0, 4m + 1]$.

- Case $b = 0$. We have $0 = 4m + 2 \equiv 2(2m + 1) \pmod{4m + 2}$.
- Case $b \in [1, 4m + 1]$ and $v_2(b) = 0$. If $b \leq 2m + 1$, then $b \in X$. If $b \geq 2m + 3$, then $q - b \in X$.
- Case $b \in [1, 4m + 1]$ and $v_2(b) = 1$. In this case, $b = 2c$, where $c \in X$.
- Case $b \in [1, 8 \left\lfloor \frac{m}{3} \right\rfloor + 4]$ and $v_2(b) = 2$. In this case, $b = 2c$, where $c \in Y$.
- Case $b \in [1, 8 \left\lfloor \frac{m}{3} \right\rfloor + 4], v_2(b) \geq 3$, and $v_2(b)$ is odd. In this case, $b \in Z$.
- Case $b \in [8 \left\lfloor \frac{m}{3} \right\rfloor + 8, 4m]$ and $v_2(b) \geq 2$. Let $b = 4\beta$, where now $\beta$ is an integer. Then

$$
4m + 2 - b = 4(m - \beta) + 2.
$$

In particular, $v_2(4m + 2 - b) = 1$. Furthermore,

$$
4m + 2 - b \leq 4m + 2 - 8 \left\lfloor \frac{m}{3} \right\rfloor - 8 \leq 4 \left\lfloor \frac{m}{3} \right\rfloor + 2,
$$

and so $4m + 2 - b \in Y$. 


Corollary 1. For all $m \geq 0$ we have

$$\frac{3m+2}{2} \leq \omega_{2,1}(4m+2) < \frac{14m+18}{9} + \left\lfloor \frac{1}{2} \log_2 \left( \left\lfloor \frac{m}{3} \right\rfloor + 1 \right) \right\rfloor.$$

Proof. The lower bound is from Theorem 9. We will show that the upper bound follows from Proposition 1. We have

$$|X| = m + 1,$$

$$|Y| = \left\lfloor \frac{m}{3} \right\rfloor + 1,$$

$$|Z| = \sum_{j \geq 1} 2^{1-2j} \left\lfloor \frac{m}{3} \right\rfloor + \frac{1}{2} < \frac{2}{3} \left\lfloor \frac{m}{3} \right\rfloor + \left\lfloor \frac{1}{2} \log_2 \left( \left\lfloor \frac{m}{3} \right\rfloor + 1 \right) \right\rfloor.$$

The first two of these are immediate.

For $|Z|$, we see that $b \in Z$ if $b = 2^{2j+1}(2\delta + 1)$ where $\delta \geq 0$, $j \geq 1$, and

$$2\delta + 1 \leq 2^{3-2j-1} \left\lfloor \frac{m}{3} \right\rfloor.$$

Hence, we must have $2^{2j} \left\lceil \frac{m}{4} \right\rceil \geq 1$, that is $2^{2j-2} \leq \left\lceil \frac{m}{4} \right\rceil$ and $2j - 2 \leq \log_2 \left( \left\lceil \frac{m}{4} \right\rceil \right)$, that is $j \leq 1 + \frac{\log_2 \left( \left\lfloor \frac{m}{3} \right\rfloor \right)}{2}$. Further, for a given $j$,

$$0 \leq \delta \leq -2^{-1} + 2^{1-2j} \left\lfloor \frac{m}{3} \right\rfloor,$$

that is, the number of $\delta$ is $2^{1-2j} \left\lfloor \frac{m}{3} \right\rfloor + \frac{1}{2}$. By Proposition 1,

$$\omega_{2,1}(4m+2) < |X| + |Y| + |Z| \leq m + 1 + \left\lfloor \frac{m}{3} \right\rfloor + 1 + \frac{2}{3} \left\lfloor \frac{m}{3} \right\rfloor + \left\lfloor \frac{1}{2} \log_2 \left( \left\lfloor \frac{m}{3} \right\rfloor + 1 \right) \right\rfloor \leq \frac{14m+18}{9} + \left\lfloor \frac{1}{2} \log_2 \left( \left\lfloor \frac{m}{3} \right\rfloor + 1 \right) \right\rfloor.$$

Another recursive construction is described next.

Construction 2. Let $S' \subseteq \mathbb{Z}_{2m+1}$ be a $(2,2;2m+1)$ covering set such that $S' \subseteq [0,m]$. Let $S = X \cup Y$, where the sets $X, Y \subseteq \mathbb{Z}_{4m+2}$ are defined by

$$X = \{2a+1 \mid a \in [0,m]\}, \quad Y = \{2s' \mid s' \in S' \setminus \{0\}.$$

Proposition 2. For all $m \geq 0$, $S$ of Construction 2 is a $(2,1;4m+2)$ covering set.
Proof. First, we see that $X$ covers 0 and all the odd elements of $\mathbb{Z}_{4m+2}$. Next, we note that the even elements of $\mathbb{Z}_{4m+2}$ are isomorphic to $\mathbb{Z}_{2m+1}$. Thus, the elements of $Y$ cover all the even non-zero elements of $\mathbb{Z}_{4m+2}$ except perhaps elements of the form $-4s'$ for $s' \in S'$. However

$$-4s' \equiv 2(2(m - s') + 1) \pmod{4m + 2},$$

and so $-4s'$ is covered by $X$ since $2(m - s') + 1 \in X$. \hfill \Box

Corollary 2. For all $m \geq 0$, $\omega_{2,1}(4m + 2) \leq m + \omega_{2,2}(2m + 1)$.

Proof. Let $S' \subseteq \mathbb{Z}_{2m+1}$ be a $(2, 2; 2m + 1)$ optimal covering set. Without loss of generality, we may assume that $S' \subseteq [0, m]$, since $s$ and $-s \equiv 2m + 1 - s \pmod{2m + 1}$ cover the same elements of $\mathbb{Z}_{2m+1}$. From Construction 2 we get

$$\omega_{2,1}(4m + 2) \leq |S| = |X| + |Y| = (m + 1) + (\omega_{2,2}(2m + 1) - 1).$$

\hfill \Box

Corollary 3 in [5] states that a $(2, 2; 2m + 1)$ perfect packing set exists if and only if $v_2(\text{ord}_p(2)) \geq 2$ for any prime $p$ dividing $2m + 1$.

Corollary 3. If $v_2(\text{ord}_p(2)) \geq 2$ for any prime $p$ dividing $2m + 1$, then $\omega_{2,1}(4m + 2) = 3m/2 + 1$, and Construction 2 produces an optimal $(2, 1; 4m + 2)$ covering set.

Proof. A simple counting argument shows that if a $(2, 2; 2m + 1)$ perfect covering set exists, then $\omega_{2,2}(2m + 1) = m/2 + 1$. We then combine Theorem 10 with Corollary 2 to obtain the desired result. \hfill \Box

Example 3. Of the first 1000 even $m$, 390 satisfy the condition of Corollary 3, the first ten are 2, 6, 8, 12, 14, 18, 20, 26, 30, 32. Of the 5000 even $m$ below 10000, 1745 satisfy the condition of Corollary 3.

References

How easy is code equivalence over \( \mathbb{F}_q \)?

*Extended Abstract

Nicolas Sendrier · Dimitris E. Simos

Received: date / Accepted: date

Abstract The linear code equivalence problem is to decide whether two linear codes over \( \mathbb{F}_q \) are identical up to a linear isometry of the Hamming space. The support splitting algorithm [25] runs in polynomial time for all but a negligible proportion of all linear codes, and solves the latter problem by recovering the isometry when it is just a permutation of the code support. While for a binary alphabet isometries are exactly the permutations, this is not true for \( q \geq 3 \). We explore in this paper, a generalization of the support splitting algorithm where we aim to retrieve any isometry between equivalent codes. Our approach is twofold; first we reduce the problem of deciding the equivalence of linear codes to an instance of permutation equivalence. To this end, we introduce the notion of the closure of a code and give some of its properties. In the aftermath, we exhibit how this algorithm can be adapted for \( q \in \{3, 4\} \), where its complexity is polynomial for almost all of its instances. Although the aforementioned reduction seems attractive, when \( q \geq 5 \) the closure reduces the instances of the linear code equivalence problem to exactly those few instances of permutation equivalence that were hard for the support splitting algorithm. Finally, we argue that for \( q \geq 5 \) the linear code equivalence problem might be hard for almost all instances.

Keywords Equivalence · Isometry · Closure of a Code · Linear Codes

Mathematics Subject Classification (2000) 94B05 · 05E20

Nicolas Sendrier\(^1\)

\(^1\) INRIA Paris-Rocquencourt, Project-Team SECRET
78153 Le Chesnay Cedex, France
E-mail: nicolas.sendrier@inria.fr

Dimitris E. Simos\(^1,2\)

\(^1\) INRIA Paris-Rocquencourt, Project-Team SECRET
78153 Le Chesnay Cedex, France
E-mail: dimitrios.simos@inria.fr

\(^2\) SBA Research, 1040 Vienna, Austria
E-mail: dsimos@sba-research.org
1 Introduction

The purpose of this work is to examine the worst-case and average-case hardness of the Linear Code Equivalence problem. That is, given the generator matrices of two $q$-ary linear codes, how hard is it to decide whether or not these codes are identical up to a linear isometry of the Hamming space? The computational version of this problem, is to retrieve the linear isometry.

The Permutation Code Equivalence problem is the restriction of the above problem when the isometries are limited to permutations of the code support\(^1\). Petrank and Roth proved [22] that the worst-case was not easier than for the Graph Isomorphism problem. On the other hand, the support splitting algorithm [25] solves the computational version of the problem in time polynomial for all but an exponentially small proportion of the instances.

For a more general notion of code equivalence which includes all linear isometries, the situation seems to change drastically. In practice, the support splitting algorithm can be extended for $q \in \{3, 4\}$, and similarly solves all but an exponentially small proportion of the instances in polynomial time. However, for any fixed $q \geq 5$, the computational and the decisional problem seems to be intractable for almost all instances.

The paper is structured as follows. In section 2, we present the different notions of code equivalence induced by isometries of the Hamming space, while in section 3, we define in formal terms all decisional and computational problems related to code equivalence and mention the most significant contributions in terms of complexity and algorithms. In section 4, we illustrate a reduction of the Linear Code Equivalence problem as an instance of the Permutation Code Equivalence, and its efficiency is analyzed in the following section. Finally, we elaborate on the hardness of these computational and decisional problems and mention possible implications, in the concluding discussion.

2 Equivalence of linear codes

Code equivalence is a basic concept in coding theory. However, the equivalence of linear codes has met a few different definitions in the literature, often without motivation. We review the concept of what it means for codes to be “essentially different” by considering the metric Hamming space together with its isometries, which are the maps preserving the metric structure. This in turn will lead to a rigorous definition of equivalence of linear codes. In fact, we will call codes isometric if they are equivalent as subspaces of the Hamming space.

Let $\mathbb{F}_q$ be a finite field of cardinality $q = p^r$, where the prime number $p$ is its characteristic, and $r$ is a positive integer. As usual, a linear $[n, k]$ code $C$ is a $k$-dimensional subspace of the finite vector space $\mathbb{F}_q^n$ and its elements are called codewords. We consider all vectors, as row vectors. Therefore, an element $v$ of $\mathbb{F}_q^n$ is of the form $v := (v_1, \ldots, v_n)$. It can also be regarded as

\(^1\) except for $q = 2$ the isometries are not limited to permutations.
the mapping \( v \) from the set \( I \) to \( \mathbb{F}_q \) defined by \( v(i) := v_i \). The Hamming distance (metric) on \( \mathbb{F}_q^n \) is the following mapping,

\[
d : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N} : (x, y) \mapsto d(x, y) := \{ i \in \{1, 2, \ldots, n\} \mid x_i \neq y_i \}.
\]

The pair \((\mathbb{F}_q^n, d)\) is a metric space, called the Hamming space of dimension \( n \) over \( \mathbb{F}_q \), denoted by \( H(n, q) \). The Hamming weight \( w(x) \) of a codeword \( x \in C \) is simply the number of its non-zero coordinates, i.e. \( w(x) := d(x, 0) \).

Two codes \( C, C' \) are of the same quality if there exists a mapping \( \iota : \mathbb{F}_q^n \to \mathbb{F}_q^n \) with \( \iota(C) = C' \) which preserves the Hamming distance, i.e.

\[
d(\iota(v), \iota(v')) = d(v, v'), \quad \text{for all } v, v' \in \mathbb{F}_q^n.
\]

Mappings with the latter property are called the isometries of \( H(n, q) \), and the two codes \( C \) and \( C' \) will be called isometric. It is well-known due to a theorem of MacWilliams that any linear \(^2\) metric between linear subspaces preserving the weight of the codewords induces an equivalence for codes \([18]\). Clearly, isometric codes have the same error-correction capabilities. We write \( S_n \) for the symmetric group acting on the set \( I \), equipped with the composition of permutations.

If \( q = p^i \) is not a prime, then the Frobenius automorphism \( \tau : \mathbb{F}_q \to \mathbb{F}_q, x \mapsto x^q \) applied on each coordinate of \( \mathbb{F}_q^n \) preserves the Hamming distance, too. Moreover, for \( n \geq 3 \), the isometries of \( \mathbb{F}_q^n \) which map subspaces onto subspaces are exactly the semilinear mappings \(^3\) of the form \((v; (\alpha, \pi))\), where 

\[(v; (\alpha, \pi)) \text{ is a linear isometry and } \alpha \text{ is a field automorphism, i.e. } \alpha \in \text{Aut}(\mathbb{F}_q) \) \text{ (c.f. [3, 14]).}

All these mappings form the group of semilinear isometries of \( H(n, q) \) which is isomorphic to the semidirect product \( \mathbb{F}_q^n \rtimes (\text{Aut}(\mathbb{F}_q) \times S_n) \), where the multiplication of elements is given by

\[
(v; (\alpha, \pi))(\varphi; (\beta, \sigma)) := (v \cdot \alpha(\varphi); (\alpha \beta, \pi \sigma)) \quad (1)
\]

where, in detail we have \((v \cdot \alpha(\varphi)_i) := v_i \alpha(\varphi_{\pi^{-1}(i)})\) for \( i = 1, \ldots, n \). Furthermore, there is a description of \( \mathbb{F}_q^n \rtimes (\text{Aut}(\mathbb{F}_q) \times S_n) \) as a generalized wreath product \( \mathbb{F}_q^n \wr (\text{Aut}(\mathbb{F}_q) \times S_n) \), see [3, 9, 14]. Clearly, the notion of semilinear isometry which can be expressed as a group action on the set of linear subspaces gives rise to the most general notion of equivalence for linear codes. The action of the latter group in an element of \( \mathbb{F}_q^n \) is translated into an equivalence for linear codes. Equivalence can also be induced by arbitrary isometries of \( H(n, q) \), but such mappings may destroy linearity and we are only interested in isometries that map linear subspaces to linear subspaces.

**Definition 1** Two linear codes \( C, C' \subseteq \mathbb{F}_q^n \) will be called semilinearly equivalent, and will be denoted as \( C \overset{\text{SLE}}{\simeq} C' \), if there exists a semilinear isometry \((v; (\alpha, \sigma)) \in \mathbb{F}_q^n \rtimes (\text{Aut}(\mathbb{F}_q) \times S_n)\) that maps \( C \) onto \( C' \), i.e.

\[
C' = (v; (\alpha, \sigma))(C) = \{ (v; (\alpha, \sigma))(x) \mid (x_i) \in C \} \text{ where } (v; (\alpha, \sigma))(x_1, \ldots, x_n) = (v_1 \alpha(x_{\pi^{-1}(1)}), \ldots, v_n \alpha(x_{\pi^{-1}(n)})).\]

\(^2\) For all \( u, v \in \mathbb{F}_q^n \) we have \( \iota(u + v) = \iota(u) + \iota(v), \iota(uv) = \iota(u) \iota(v) \) and \( \iota(0) = 0 \).

\(^3\) \( \sigma : \mathbb{F}_q \to \mathbb{F}_q \) is semilinear if there exists \( \alpha \in \text{Aut}(\mathbb{F}_q) \) such that for all \( u, v \in \mathbb{F}_q^n \) and \( k \in \mathbb{F}_q \) we have \( \sigma(u + v) = \sigma(u) + \sigma(v) \) and \( \sigma(ku) = \alpha(k)\sigma(u) \).

\(^4\) The action of the semilinear and linear group in an element of \( \mathbb{F}_q^n \) can be seen at definitions 1 and 2, respectively.
The group of semilinear isometries of $H(n, q)$ reduces to the group of linear isometries if and only if $q$ is a prime (since $\text{Aut}(\mathbb{F}_q^n)$ is trivial if and only if $q$ is a prime). The latter group corresponds to the semidirect product of $\mathbb{F}^*_q \wr S_n$, $\mathbb{F}^*_q \rtimes S_n = \{(u; \pi) \mid u : I_n \mapsto \mathbb{F}^*_q, \pi \in S_n\}$, called the monomial group of degree $n$ over $\mathbb{F}^*_q$. Note that, some authors [3,8,10], describe this group as the wreath product $\mathbb{F}^*_q \rtimes \mathcal{S}_n$. Therefore, by restricting the group of semilinear isometries to the group of linear isometries we have another notion of equivalence for linear codes.

**Definition 2** Two linear codes $C, C' \subseteq \mathbb{F}^*_q$ will be called linearly or monomially equivalent, and will be denoted as $C \equivLE C'$, if there exists a linear isometry $\iota = (u; \sigma) \in \mathbb{F}^*_q \wr S_n$ that maps $C$ onto $C'$, i.e. $C' = (u; \sigma)(C) = \{(u; \sigma)(x) \mid (x_1, \ldots, x_n) \in C\}$ where $(u; \sigma)(x_1, \ldots, x_n) := (u_1 x_{\sigma^{-1}(1)}, \ldots, u_n x_{\sigma^{-1}(n)})$.

In addition, when $\mathbb{F}_q = \mathbb{F}_2$ the group of linear isometries of $H(n, 2)$ is isomorphic to $S_n$, and these isometries correspond to permutation of coordinates.

**Definition 3** Two linear codes $C, C' \subseteq \mathbb{F}^*_q$ will be called permutationally equivalent and will be denoted as $C \equivPE C'$, if there exists a permutation $\sigma \in S_n$ that maps $C$ onto $C'$, i.e. $C' = \sigma(C) = \{\sigma(x) \mid x = (x_1, \ldots, x_n) \in C\}$ where $\sigma(x) = \sigma(x_1, \ldots, x_n) := (x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(n)})$.

Moreover, there is a particular subgroup of $S_n$ that maps $C$ onto itself, the permutation group of $C$ defined as $\text{PAut}(C) := \{C = \sigma(C) \mid \sigma \in S_n\}$. $\text{PAut}(C)$ always contains the identity permutation. If it does not contain any other element, we will say that it is trivial. Finally, we can define the monomial group of $C$ as $\text{MAut}(C) := \{C = (u; \sigma)(C) \mid (u; \sigma) \in \mathbb{F}^*_q \wr S_n\}$ and the automorphism group of $C$ as $\text{Aut}(C) := \{C = (u; \sigma)(C) \mid (u; \sigma) \in \mathbb{F}^*_q \wr \text{Aut}(\mathbb{F}_q \times S_n)\}$ where their elements map each codeword of $C$ to another codeword of $C$, under the respective actions of the involved groups. For more details, on automorphism groups of linear codes we refer to [13].

### 3 Previous work

For efficient computation of codes we represent them with generator matrices. A $k \times n$ matrix $G$ over $\mathbb{F}_q$, is called a generator matrix for the $[n, k]$ linear code $C$ if the rows of $G$ form a basis for $C$, so that $C = \{xG \mid x \in \mathbb{F}^*_q\}$. In that case, we denote the code $C$ that is spanned by the generator matrix $G$, as $C = \langle G \rangle$.

In general, a linear code possess many different bases, and it is clear from linear algebra that the set of all generator matrices for $C$ can be reached by $\{SG \mid S \in \text{GL}_k(q)\}$, where $\text{GL}_k(q)$ is the group of all $k \times k$ invertible matrices over $\mathbb{F}_q$.

Since every linear code can be represented with a generator matrix, we express the equivalence between linear codes in terms of their generator matrices. As we have three different notions of equivalence, we define the respective decisional problems, below. The first one is w.r.t. the semilinear equivalence.
Problem 1 (Semilinear Code Equivalence (SLCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Question: are \(\langle G \rangle_{\text{SLE}} \sim \langle G' \rangle\)?

In a similar manner, we can define decisional problems related to linear and permutation equivalence.

Problem 2 (Linear Code Equivalence (LCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Question: are \(\langle G \rangle_{\text{LE}} \sim \langle G' \rangle\)?

Problem 3 (Permutation Code Equivalence (PCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Question: are \(\langle G \rangle_{\text{PE}} \sim \langle G' \rangle\)?

The computational versions of all three previous decisional problems, is to retrieve the equivalence mapping between the codes. Again, we begin with the semilinear equivalence.

Problem 4 (Computational Semilinear Code Equivalence (CSLCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Problem: Find a semilinear isometry \((\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^* \circledast (\text{Aut} (\mathbb{F}_q) \times S_n)\) such that \(\langle G' \rangle = (\upsilon; (\alpha, \sigma)) (\langle G \rangle)\).

Finally, we define the computational versions of the LCE and PCE problems.

Problem 5 (Computational Linear Code Equivalence (CLCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Problem: Find a linear isometry \((\upsilon; \sigma) \in \mathbb{F}_q^* \circledast S_n\) such that \(\langle G' \rangle = (\upsilon; \sigma) (\langle G \rangle)\).

Problem 6 (Computational Permutation Code Equivalence (CPCE))

Parameters: \(n, k, q\).
Instance: two matrices \(G, G' \in \mathbb{F}_q^{k \times n}\).
Problem: Find a permutation \(\sigma \in S_n\) such that \(\langle G' \rangle = \sigma (\langle G \rangle)\).

One of our goals is to explore the hardness of the LCE and CLCE problems, therefore we deem necessary to briefly mention the most significant results in terms of complexity, for deciding them, and algorithms, for computing them.
3.1 Past complexity results

The PCE problem, was introduced in [22], who showed that if $\mathbb{F}_q = \mathbb{F}_2$ then it is harder than the Graph Isomorphism (GI) problem, there exists a polynomial time reduction, but not NP-complete unless $P = NP$. A different proof of this reduction is also given in [14]. Recently, the reduction of [22] was generalized in [12] over any field $\mathbb{F}_q$, hence the PCE problem is harder than the GI problem, for any field $\mathbb{F}_q$. The latter problem, has been extensively studied for decades, but until now there is no polynomial-time algorithm for solving all of its instances.

Last but not least, we would like to mention that the McEliece public-key cryptosystem [19] is related to the hardness of permutationally equivalent binary linear codes. Towards this direction, another important complexity result was shown in [6], that the Hidden Subgroup problem also reduces to the PCE problem for any field $\mathbb{F}_q$.

3.2 Related algorithms for code equivalence

Due to its relation to the GI problem, some researchers have tried to solve the CPCE problem by interpreting graph isomorphism algorithms to codes. This approach, was followed in [5] using the fact that orbits under edge local complementation of a bipartite graph correspond to equivalence classes of binary linear codes. Mapping codes to graphs and using the software Nauty by B. D. McKay has been used in [20], for binary, ternary and quaternary codes where the permutation, linear and semi-linear equivalence was considered, respectively. Moreover, an adaptation of Luks's algorithm for hypergraph isomorphism for solving the CPCE problem over any $\mathbb{F}_q$ was presented in [2], whose complexity is simply-exponential in the length $n$ of a code $C \subseteq \mathbb{F}_q^n$. Another approach using bipartite graphs for the CLCE problem over small fields was given in [4], where code equivalence is reduced to a decision problem regarding isomorphism of binary matrices. Note also, that in this work also the semi-linear equivalence was considered for $\mathbb{F}_4$. Computation of canonical forms for generator matrices of linear codes for the CSLCE problem over $\mathbb{F}_q$ by formulating the equivalence classes of codes as orbits of a group action from the left on the set of generator matrices was given in [7]. It is worthwhile also to mention the algorithm of J. Leon for computing the automorphism group of a code [15], which is available for many computer algebra systems like GAP and MAGMA, and is used also for testing code equivalence. More specifically, in GAP, it is implemented for solving the CPCE problem over the binary field, while in MAGMA the implementation works for the CLCE problem, for small prime fields and for $\mathbb{F}_4$. However, Leon’s algorithm requires a time exponential in the code dimension since it computes the set of all codewords of minimum weight.

Finally, we would like to remark that, to the best of our knowledge there is no efficient algorithm for solving the CLCE problem for any field $\mathbb{F}_q$. 
The support splitting algorithm can be used as an oracle to decide whether two binary codes are permutationally equivalent [25], as well as to retrieve the equivalence mapping. The main idea is to partition the support $I_n$ of a code $C \subseteq \mathbb{F}_2^n$ into small sets that are fixed under operations of $\text{PAut}(C)$. The algorithm employs the concept of invariants and signatures, defined below.

Let $L_{n,k}$ denote the set of all linear codes of length $n$ and dimension $k$, and let $L = \bigcup_{n,k>0} L_{n,k}$ be the set of all such codes.

**Definition 4** An invariant $R$ over a set $E$ is defined to be a mapping $R : L \mapsto E$ such that any two permutation equivalent codes take the same value, i.e. if $C \overset{\text{PE}}{\sim} C' \implies R(C) = R(C')$.

For instance, the Hamming weight enumerator is an invariant over the polynomials with integer coefficients. Applying an invariant, for instance the weight enumerator, may help us to decide whether two codes are equivalent or not.

**Definition 5** A signature $S$ over a set $F$ maps a code $C \subseteq F^n$ and an element $i \in I_n$ into an element of $F$ and is such that for all $\sigma \in S_n$, $S(C, i) = S(\sigma(C), \sigma(i))$. Moreover, $S$ is called discriminant for $C$ if there exist $i, j \in I_n$ such that $S(C, i) \neq S(C, j)$ and fully discriminant if this holds $\forall i, j \in I_n$.

If $S$ is fully discriminant for $C$, and $C' = \sigma(C)$ for $\sigma \in S_n$, we are able to retrieve $\sigma$. The support splitting algorithm (SSA) takes as an argument a generator matrix $G$ for a code $C$ and returns a labeled partition $I = \{(I_j, j)\}_{j \in I_n}$ of the code support. For any two linear codes $C$ and $C'$ with generator matrices $G$ and $G'$, let $SSA(G) = \{(I_j, j)\}_{j \in I_n}$ and $SSA(G') = \{(I'_j, j)\}_{j \in I_n}$.

The fundamental property of $SSA$ is that if

$$C' = \sigma(C) \implies \forall j \in I_n, \quad I'_j = \sigma(I_j)$$

and implies in particular that the output of $SSA$ is independent of the choice of $G$. The converse of relation (2) is not necessarily true, but satisfied in practice under the assumption that the cells of the output of $SSA$ achieve the orbits of the elements of the code support w.r.t. the action of $\text{PAut}(C)$ and constitute its finest obtainable partition [17, 25]. The main difficulty of the algorithm, is to obtain a fully discriminant signature, for as many codes as possible. In [25] it was shown that such a signature, can be built from the weight enumerator of the hull of a code $C$, denoted by $H(C)$, and defined as the intersection of the code with its dual, $H(C) = C \cap C^\perp$ [1], because the hull commutes with permutations\(^5\), $H(\sigma(C)) = \sigma(H(C))$, and therefore it is an invariant for permutation equivalence. The (heuristic) complexity of $SSA$ for an $[n, k]$ code $C$ is $O(n^3 + 2^h n^2 \log n)$ where $h$ is the dimension of the hull [21, 25]. In practice, for random codes, the hull has a small dimension with overwhelming probability [24] and the dominant cost for the average case is $O(n^3)$. Note that, the worst case occurs when the hull dimension is maximal; weakly self-dual codes $(C \subseteq C^\perp)$ are equal to their hulls. Then the algorithm becomes intractable with a complexity equal to $O(2^h n^2 \log n)$.

\(^5\) No such property exists in general for linear codes when (semi)-linear equivalence is considered, see also lemma 2.
3.3 Computational vs. decisional code equivalence

The purpose of this section is to exhibit the relation of the worst-case complexities of the CPCE, CLCE and CSLCE problems. If one can explicitly compute the latter problems then one can also solve its corresponding decisional versions. We argue that the other direction is also possible, that is provided we have access to an oracle for deciding the PCE, LCE and SLCE problems we can build an algorithm for computing the CPCE, CLCE and CSLCE problems, respectively. Therefore, the computational and decisional problems related to code equivalence belong in the same complexity class.

In the case of the PCE problem, the oracle used is an abstract version of SSA denoted by \( \text{Or}_{\text{PCE}}(G, G') \in \{\text{True}, \text{False}\} \). This oracle takes as input two generator matrices \( G \) and \( G' \) of two \( q \)-ary linear codes and is ideal, in the sense that it always return true if the generator matrices span permutationally equivalent codes, and false otherwise. It is well known, that punctured codes of equivalent codes remain equivalent, when the first are punctured in the same position. For our result, we impose a stronger condition and state without proof the following lemma.

**Lemma 1** Let \( G \) and \( G' \) span two \([n, k]\) linear codes \( C \) and \( C' \) over \( \mathbb{F}_q \). If \( \text{Or}_{\text{PCE}}(G, G') \) is True and \( \text{Or}_{\text{PCE}}(G_i, G'_j) \) is True for some \( i, j \in \mathbb{I}_n \) then there exists \( \sigma \in S_n \) such that \( C' = \sigma(C) \) and \( j = \sigma(i) \).

The previous lemma is the keystone for proving that the computational and decisional version of the permutation code equivalence are equally hard. A similar lemma can be stated for the LCE or SLCE problem where the previous oracle can be used as a building block of an algorithm that retrieves the permutational part of the linear or semilinear isometry of the CLCE and CSLCE problems. Hence, the computational and decisional problems of code equivalence are not essentially different. Now, consider an algorithm that solves the CSLCE or CLCE problem to which is given an instance of the CPCE problem. Due to the previous discussion, the expected output of the algorithm, a linear or semilinear isometry, is just a permutation.

4 Reduction of linear code equivalence to permutation code equivalence

Hence, we have at our disposal an algorithm, the support splitting algorithm, that solves the PCE and CPCE problems in (almost) polynomial time. Therefore, it is natural to investigate a reduction of the LCE problem as an instance of the PCE problem. To this end, we introduce the closure of a linear code.

**Definition 6** Let \( \mathbb{F}_q = \{a_0, a_1, \ldots, a_{q-1}\} \), with \( a_0 = 0 \), and a linear code \( C \subset \mathbb{F}_q^n \). Define \( \mathcal{I}_{q-1}^{(n)} \) as the cartesian product of \( \mathcal{I}_{q-1} \times \mathcal{I}_n \). The closure \( \tilde{C} \) of
How easy is code equivalence over $\mathbb{F}_q$?

the code $C$ is a code of length $(q - 1)n$ over $\mathbb{F}_q$ where,

$$\tilde{C} = \{(a_kx_i)_{(k,i)\in I_{q-1}^n} \mid (x_i)_{i \in I_n} \in C\}.$$  

Clearly, we see that every coordinate of the closure $\tilde{C}$, corresponds to a coordinate position of a codeword of $C$ multiplied by a nonzero element of $\mathbb{F}_q$. Since, the index $(k,i) \in I_{q-1}^n$ of a position of a codeword of the closure means that $k \in I_{q-1}$ and $i \in I_n$, we have taken into account every possible multiplication of $x_i$ with nonzero elements of $\mathbb{F}_q$, and it is easy for someone to show\(^6\) the following:

**Theorem 1** Let $C, C' \subseteq \mathbb{F}_q^n$. If $C$ and $C'$ are linearly equivalent, i.e. $C \overset{\text{LE}}{\sim} C'$ then $\tilde{C}$ and $\tilde{C}'$ are permutationally equivalent, i.e. $\tilde{C} \overset{\text{PE}}{\sim} \tilde{C}'$.

Theorem 1 is of great importance, because it realizes a reduction from the LCE problem to the PCE problem. Thus, we are able to decide if the codes $C$ and $C'$ are linearly equivalent by checking their closures for permutation equivalence. Moreover, if the closures are permutation equivalent there might be an algorithmic procedure that will allow us to recover the initial isometry between $C$ and $C'$. However, as we shall see shortly after, the closure reduces an instance of the CLCE problem to exactly those instances that were hard for the support splitting algorithm for tackling the CPCE problem over $\mathbb{F}_q$, $q \geq 5$.

We would also like to mention that this representation of the closure is not unique. In particular, it depends on a lexicographical ordering of $\mathbb{F}_q^n$.

For example, the ordering $(a_1,1) < \ldots < (a_1,n) < (a_2,1) < \ldots < (a_2,n) < \ldots < (a_{q-1},1) < \ldots < (a_{q-1},n)$ gives a total order for $I_{q-1}^n$, and gives rise to the following closure,

$$\tilde{C} = \{(a_1x_1, \ldots, a_1x_n, \ldots, a_{q-1}x_1, \ldots, a_{q-1}x_n) \mid (x_1, \ldots, x_n) \in C\}.$$  

Note that, such an ordering can always be induced by a permutation of the symmetric group $\mathcal{S}_{\mathbb{F}_q}$ acting on $\mathbb{F}_q$ defined as $\mathcal{S}_{\mathbb{F}_q} := \{\rho \mid \rho : \mathbb{F}_q \to \mathbb{F}_q, \rho \text{ is a bijection and } \rho(0) = 0\}$.

Moreover, it is natural to ask which permutations can appear as permutations of the closures since $SSA$ was designed exactly to retrieve the permutation between equivalent codes. If we assume that we are given a primitive element $\gamma$ of $\mathbb{F}_q$, it is well-known that all of its permissible powers generate the multiplicative group of $\mathbb{F}_q = \{\gamma, \gamma^2, \ldots, \gamma^{q-2}, \gamma^{q-1} = 1\}$. Then an ordering according to a cyclic shift of a power of $\gamma$ will produce a unique closure for the code $C$ (consider the row echelon form on two generator matrices of the closures produced by such orderings).

Since, such a closure can always be reached by a composition of permutations of $\mathcal{S}_{\mathbb{F}_q}$, we define a systematic form for the closure as follows,

---

\(^6\) The detailed proof of this theorem and all subsequent results will appear in an extended version of this paper.
\[ \tilde{C}_{\text{sys}} = \{(x_1, \gamma x_1 \ldots, \gamma^{q-2} x_1, \ldots, x_n, \gamma x_n, \ldots, \gamma^{q-2} x_n) \mid (x_1, \ldots, x_n) \in C\}. \]

If we consider the cyclic group \( C_{q-1} \) of order \( q - 1 \) there is a natural isomorphism between \( \mathbb{F}_q^n \times S_n \) and \( C_{q-1} \triangleleft S_n \), the semidirect product of \( n \) copies of \( C_{q-1} \) and \( S_n \), called also the generalized symmetric group and denoted by \( S(q - 1, n) \). Its order is \( (q - 1)^n n! \) and its elements are exactly those permutations that can appear as permutations of permutationally equivalent closures. This reasoning is sufficient for one to show that for \( C \subseteq \mathbb{F}_q^n \) we have \( \text{MAut}(C) \) to be isomorphic to \( \text{PAut}(\tilde{C}) \cap S(q - 1, n) \).

Moreover, it further implies that the converse of theorem 1 also holds, and by involving the systematic forms of the closures as an intermediate step, after a non-trivial proof, we can show the following relation for equivalent codes and their closures.

**Theorem 2** Let \( C, C' \subseteq \mathbb{F}_q^n \). Then \( C \) and \( C' \) are linearly equivalent, i.e. \( C \overset{\text{LE}}{\sim} C' \), if and only if \( \tilde{C} \) and \( \tilde{C}' \) are permutationally equivalent, i.e. \( \tilde{C} \overset{\text{PE}}{\sim} \tilde{C}' \).

### 5 Efficiency of the reduction

The \( SSA \) used as an invariant the hull \( H(C) \) of a code. In order to explore possible extensions of \( SSA \) we have to determine the quality of the hull of the closure \( H(\tilde{C}) = \tilde{C} \cap \tilde{C}^\perp \), where the dual of the closure is defined according to some inner product. We consider two inner products, the Euclidean and Hermitian inner product, defined below:

- \( \langle x, y \rangle_E = \sum_{i=1}^n (x_i, y_i)_E = \sum_{i=1}^n x_i y_i = x_1 y_1 + \ldots + x_n y_n \in \mathbb{F}_q \). If \( q \) is a square, \( \langle x, y \rangle_H \) (below) is generally preferred to \( \langle x, y \rangle_E \).
- \( \langle x, y \rangle_H = \sum_{i=1}^n (x_i, y_i)_H = \sum_{i=1}^n x_i y_i = x_1 \bar{y}_1 + \ldots + x_n \bar{y}_n \in \mathbb{F}_q \), where \( q \) is an even power of a prime with \( \pi = x \sqrt{q} \) for \( x \in \mathbb{F}_q \) (cf. [23]). Note that, for \( x, y \in \mathbb{F}_q \),
  \[ (x + y)\sqrt{q} = x \sqrt{q} + y \sqrt{q}, \quad x^q = x. \]

Now, consider two codewords \( \tilde{x}, \tilde{y} \) of the closure \( \tilde{C} \) of \( C \subseteq \mathbb{F}_q^n \). Then their Euclidean and Hermitian inner product is given by \( \langle \tilde{x}, \tilde{y} \rangle_E = \left( \sum_{i=1}^{q-1} a_i^2 \right) \langle x, y \rangle_E \) and \( \langle \tilde{x}, \tilde{y} \rangle_H = \left( \sum_{i=1}^{q-1} a_i \pi_i \right) \langle x, y \rangle_H \), respectively, where \( \mathbb{F}_q = \{ a_0 = 0, a_1, \ldots, a_{q-1} \} \). Using lemma 7.3. of [16] which states that \( a_0, a_1, \ldots, a_{q-1} \) are distinct if and only if \( \sum_{i=0}^{q-1} a_i t = 0 \) for \( t = 0, 1, \ldots, q - 2 \) and \( \sum_{i=0}^{q-1} a_i = -1 \) for \( t = q - 1 \), we can show that,

\[
\langle \tilde{x}, \tilde{y} \rangle_E = \begin{cases} 
0 & \text{for } q \geq 4 \\
- \langle x, y \rangle_E & \text{for } q = 3
\end{cases} \quad \text{and} \quad \langle \tilde{x}, \tilde{y} \rangle_H = \begin{cases} 
0 & \text{for } q \geq 4 \\
- \langle x, y \rangle_H & \text{for } q = 4.
\end{cases}
\]
How easy is code equivalence over $\mathbb{F}_q$?

This means, that the closure $\tilde{C}$ is a weakly self-dual code for every $q \geq 5$, considering both Euclidean and Hermitian duals, which is exactly the hard instances of SSA. Moreover, for $\mathbb{F}_3$ and $\mathbb{F}_4$ equipped with the Euclidean and Hermitian inner product, respectively, the distribution of the dimension of $\mathcal{H}(\tilde{C})$ follows the distribution of the dimension $\mathcal{H}(C)$, since the closure has the same dimension as $C$, and will be on average a small constant, $[24]$, except in the cases where $C$ is also a weakly self-dual code.

It is worth mentioning that these are exactly the same cases where the hull of a code could be used as an invariant for (semi)-linear equivalence, because the duals of linear and semilinear codes remain equivalent with the same isometry of the original codes only in $\mathbb{F}_3$ and $\mathbb{F}_4$, due to the following relation (see $[13,25]$):

**Lemma 2** Let $C \subseteq \mathbb{F}_q^n$, and $(v; (\sigma, \alpha)) \in \mathbb{F}_q^n \times (\text{Aut}(\mathbb{F}_q) \times S_n)$. Then

(i) $(v; (\sigma, \alpha))(C) = ((v^{-1}; (\sigma, \alpha))(C)^\perp)$ where $C^\perp$ is w.r.t. $\langle, \rangle_E$.

(ii) $(v; (\sigma, \alpha))(C) = ((v^{-1}; (\sigma, \alpha))(C)^\perp)$ where $C^\perp$ is w.r.t. $\langle, \rangle_H$.

Then, a signature for an extension of SSA can be built from the weight enumerator of the $\mathcal{H}(\tilde{C})$. The LCE and CLCE problems can be decided (and computed) in polynomial time using SSA only in $\mathbb{F}_3$ and $\mathbb{F}_4$, as long as the hull of the given code is small (the worst-case being a weakly self-dual code). It does not seem possible to extend this result to larger alphabet. We conclude by posing the following conjecture.

**Conjecture 1** For a given $q \geq 5$, the LCE and CLCE problems over $\mathbb{F}_q$ are hard for almost all instances.

Note that, there is a similar negative complexity result due to Dirk Vertigan $[27]$. The result is given for graphs, but, translated for codes, it states that evaluating the (homogeneous) weight enumerator polynomial of a linear code over $\mathbb{F}_q$ for $q \geq 5$ on any point of the complex unit circle is always difficult except for a constant number of trivial points. The evaluation of the weight enumerator in those points essentially provide the code cardinality. There is an additional point easy to evaluate for $q \in \{2, 3, 4\}$. The evaluation in this point essentially provides the cardinality of the hull of the code. For $q = 4$ the hull is defined according to the hermitian inner product. There is possibly more than just a coincidence here, but the connection with code equivalence is not obvious to establish. Doing so would certainly be enlightening.

**6 Conclusion**

In this paper, we explored the hardness of the (Computational) Linear Code Equivalence problem(s) over $\mathbb{F}_q$. We showed that an extension of SSA for solving the latter problems when $q \in \{3, 4\}$ is possible, in (almost) polynomial time, however for $q \geq 5$ its complexity growth becomes exponential for all instances. Moreover, we conjectured that, for $q \geq 5$, the computational and decisional version of linear code equivalence are hard for almost
all instances. Our argument, is supported by some impossibility results on the Tutte polynomial of a graph which corresponds to the weight enumerator of a code. On the bright side, the negativity of our claim, might lead to some interesting features for applications. For example, in cryptography, zero-knowledge protocols have been designed in the past, based on the hardness of the Permutation Code Equivalence problem [11]. Moreover, the relation of the automorphism groups of the code and its closure might be of cryptographic interest. The context of the framework built in [6] suggests that codes with large automorphism groups resist quantum Fourier sampling as long as permutation equivalence is considered. It would thus be intriguing to investigate, if this result can also be extended for the linear and semilinear code equivalence.

Acknowledgements The authors are grateful to the reviewers for their comments and suggestions that improved the quality and the presentation of the paper. The work of the second author was carried out during the tenure of an ERCIM “Alain Bensoussan” Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 246016.

References

How easy is code equivalence over $\mathbb{F}_q$?

Exponents of polar codes using algebraic geometric code kernels

Sarah E. Anderson · Gretchen L. Matthews

Received: date / Accepted: date

Abstract Reed-Solomon and BCH codes were considered as kernels of polar codes by Mori and Tanaka in [6] and Korada, Şaşoğlu, and Urbanke in [4] to create polar codes with large exponents. For codes of length $l \leq q$, Mori and Tanaka showed Reed-Solomon codes give the best possible exponent, where $q$ is the size of the input alphabet. However, considering larger matrices over $\mathbb{F}_q$ yields better exponents for longer codes. We expand on these ideas by using AG codes as kernels of polar codes. We provide a lower bound on the exponent of such kernels. We demonstrate that both Hermitian and Suzuki kernels have larger exponents than comparable Reed-Solomon codes, for $q > 4$.

Keywords Polar codes · kernel · exponent · maximal function fields · algebraic geometric codes

1 Introduction

Polar codes were developed by Arikan [1] as an explicit construction of symmetric capacity achieving codes for binary DMCs with low encoding and decoding complexity. Polar codes were generalized to arbitrary channels by Şaşoğlu, Telatar, and Arkan [8]. In this work, we consider a $q$-ary DMC $W : \mathcal{X} \rightarrow \mathcal{Y}$ with input alphabet $\mathcal{X} := \mathbb{F}_q$, the finite field with $q$ elements where $q$ is a prime power, and transition probabilities $W(y|x)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Combining
and splitting is used to produce a block of channels as follows. First, $N$ independent copies of $W$ are combined into a channel $W_N : \mathcal{X}^N \rightarrow \mathcal{Y}^N$ defined by the transition probabilities

$$W_N(y_1^N | u_1^N) = W_N(y_1^N | u_1^N(B_N G_2^\otimes n)) = \prod_{i=1}^{N} W(y_i | u'_i),$$

for $u_1^N \in \mathcal{X}^N$ and $y_1^N \in \mathcal{Y}^N$ where the kernel matrix is

$$G_2 := \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

$\otimes$ denotes the Kronecker power, $B_N$ is an $N \times N$ bit-reversal permutation matrix, and $u'_i$ denotes the $i^{th}$ entry of $u_1^N(B_N G_2^\otimes n)$. Then $W_N$ is “split” into $N$ channels $W_N^{(i)} : \mathcal{X} \rightarrow \mathcal{Y}^N \times \mathcal{X}^{i-1}$, $1 \leq i \leq N$, such that

$$W_N^{(i)}(y_1^N, u_1^{i-1} | u_i) := \sum_{u_{i+1},...,u_N \in \mathcal{X}^{N-i}} \frac{1}{q_{N-i}} W_N(y_1^N | u_1^N)$$

for $y_1^N, u_1^{i-1} \in \mathcal{Y}^N \times \mathcal{X}^{i-1}$ and $u_i \in \mathcal{X}$.

For fixed $\delta$, as $N$ goes to infinity, the fraction of channels $\{W_N^{(i)}\}$ such that $I(W_N^{(i)}) \in (1 - \delta, 1]$ approaches $I(W)$ and the fraction of $\{W_N^{(i)}\}$ such that $I(W_N^{(i)}) \in [0, \delta)$ approaches $1 - I(W)$, where $1 \leq i \leq N$ and $I(W)$ is the highest rate at which reliable communication is possible; this phenomenon is known as polarization.

The rate at which these channels polarize is related to the kernel matrix. Since [1], matrices other than $G_2$ have been considered as kernels. Let $G$ be a $l \times l$ matrix over $\mathbb{F}_q$. Then a block of $N = l^n$ channels is produced. Here, the channel $W_N : \mathcal{X}^N \rightarrow \mathcal{Y}^N$ is defined by

$$W_N(y_1^N | u_1^N) = W_N(y_1^N | u_1^N(B_N G^\otimes n))$$

where $B_N$ is an $N \times N$ bit-reversal permutation matrix that sends $u_1^N$ to

$$(u_1, u_{l+1}, \ldots, u_{N-(l-1)}), u_2, u_{l+2}, \ldots, u_{N-(l-2)}, \ldots, u_l, u_{2l}, \ldots, u_{N}).$$

Again, $W_N$ is split into $N$ channels $W_N^{(i)} : \mathcal{X} \rightarrow \mathcal{Y}^N \times \mathcal{X}^{i-1}$, $1 \leq i \leq N$, which are defined by the transition probabilities

$$W_N^{(i)}(y_1^N, u_1^{i-1} | u_i) := \sum_{u_{i+1},...,u_N \in \mathcal{X}^{N-i}} \frac{1}{q_{N-i}} W_N(y_1^N | u_1^N).$$

It is important to note that not all $l \times l$ matrices polarize. In fact, [4] shows that polarization occurs in the case of a binary DMC if the kernel $G$ is a non-upper triangular matrix; [5] demonstrates that polarization takes place for a $q$-ary DMC provided $G = (G_{ij})$ is a non-diagonal, linear kernel, and at least one entry $G_{ij}$ with $i > j$ is a primitive element of $\mathbb{F}_q$. The rate of polarization of a kernel is known as the exponent, which is defined below.
Definition 1 [5] For any $q$-ary DMC $W$ with $0 < I(W) < 1$, an $l \times l$ kernel $G$ has rate of polarization $E(G)$, called the exponent of $G$, if and only if

- for any fixed $\beta < E(G)$,
  \[ \liminf_{n \to \infty} \Pr[Z_n \leq 2^{-ln^\beta}] = I(W), \text{ and} \]
- for any fixed $\beta > E(G)$,
  \[ \liminf_{n \to \infty} \Pr[Z_n \geq 2^{-ln^\beta}] = 1, \]

where $\{Z_n| n \geq 0\} = \{Z(W_n)| n \geq 0\}$ and $Z(W_n)$ is the Bhattacharyya parameter of $W_n$. The $W_i$ are defined recursively as

\[ W_0 = W, \text{ and } W_{n+1} = (W_n)^{(B_n+1)}, \]

where $\{B_n| n \geq 1\}$ is a sequence of i.i.d random variables uniformly distributed over the set $\{1, \ldots, l\}$.

It follows that for any fixed rate $0 < R < I(W)$ and $0 < \beta < E(G)$, there exist a sequence $\{A_N\}$ of sets $A_N \subseteq \{1, \ldots, N\}$ such that $|A_N| \geq NR$ and

\[ \sum_{i \in A_N} Z(W_N^{(i)}) = o(2^{-ln^\beta}). \]

Information is then sent across the channels indexed by elements of $A_N$.

The exponent of a matrix can be found using partial distances. The partial distances, $D_i$, $i = 1, \ldots, l$, of an $l \times l$ matrix $G = [g_1^T, \ldots, g_l^T]^T$ over $F_q$ are

\[ D_i := d(g_i, \langle g_{i+1}, \ldots, g_l \rangle), \]

where $d$ denotes Hamming distance and $\langle g_{i+1}, \ldots, g_l \rangle$ is the $F_q$-vector space spanned by $g_{i+1}, \ldots, g_l$.

Lemma 1 [4] [5] If $G$ is an $l \times l$ matrix, then the exponent of the polar code with kernel $G$ is

\[ E(G) = \frac{1}{l} \sum_{i=1}^{l} \log(D_i). \]

According to Lemma 1, Arikan’s original kernel matrix $G_2$ has exponent $E(G_2) = \frac{1}{2}$. In [4], a matrix obtained from a generator matrix for a shortened BCH code is found to have exponent greater than $\frac{1}{2}$. This, together with the above lemma, leads one to consider kernels that are generator matrices of linear codes. The partial distances of the kernel may then be estimated by bounds on the minimum distances of the nested codes. As we see in the next section, algebraic geometric codes lend themselves naturally to this construction.
2 Algebraic geometric codes

Let $F$ be a function field over $\mathbb{F}_q$ of genus $g$. An algebraic geometric (AG) code $C(D, A)$ is constructed using divisors $A$ and $D = P_1 + \cdots + P_n$ on $F$ with disjoint support, where the $P_i$ are distinct places of $F$ of degree 1. In fact,

$$C(D, A) = \{ (f(P_1), \ldots, f(P_n)) | f \in \mathcal{L}(A) \} \subseteq \mathbb{F}_q^n,$$

where

$$\mathcal{L}(A) = \{ f \in F | (f) \geq -A \} \cup \{0\}$$

is the Riemann-Roch space of $A$. If $|\text{supp}(A)| = 1$, then $C(D, A)$ is called a one-point code; otherwise, $C(D, A)$ is known as a multipoint code. If $\{f_1, \ldots, f_k\}$ is a basis for $\mathcal{L}(A)$, then

$$G = \begin{bmatrix}
    f_k(P_1) & f_k(P_2) & \cdots & f_k(P_n) \\
    f_{k-1}(P_1) & f_{k-1}(P_2) & \cdots & f_{k-1}(P_n) \\
    \vdots & \vdots & \ddots & \vdots \\
    f_1(P_1) & f_1(P_2) & \cdots & f_1(P_n)
\end{bmatrix}$$

is a generator matrix for $C(D, A)$. Note that $C(D, A)$ is an $[n, k, d]$ code where $d \geq n - \deg(A)$ and

$$k = l(A) - l(A - D) = \deg(A) + 1 - g$$

if $2g - 1 \leq \deg(A) \leq n$ and $l(A) := \dim(\mathcal{L}(A))$. Moreover, $C(D, A)$ satisfies a Singleton-like bound

$$n + 1 - g \leq k + d \leq n + 1.$$ 

An especially useful property of AG codes is their “nested” structure. If $A = \sum_{Q_i \in \mathbb{P}_F} a_i Q_i$ and $B = \sum_{Q_i \in \mathbb{P}_F} b_i Q_i$, where $\mathbb{P}_F$ is the set of all places of $F/\mathbb{F}_q$, then $A \leq B$ if $a_i \leq b_i$ for all $i$. Given divisors $A$ and $B$ with $\text{supp}(A) \cap \text{supp}(D) = \emptyset = \text{supp}(B) \cap \text{supp}(D)$,

$$A \leq B \Rightarrow \mathcal{L}(A) \subseteq \mathcal{L}(B),$$

which in turn implies $C(D, A) \subseteq C(D, B)$. In the next section, we employ AG codes as kernels of polar codes.

3 Construction of kernels using AG codes

Let $F/\mathbb{F}_q$ be a function field of genus $g$ and $P_1, \ldots, P_n$ be places of $F$ of degree one. Construct a sequence of divisors $A_1 \leq \cdots \leq A_n$ so that the supports of $D := P_1 + \cdots + P_n$ and the $A_j$, $1 \leq j \leq n$, are disjoint and

$$C(D, A_1) \subsetneq C(D, A_2) \subsetneq \cdots \subsetneq C(D, A_n) = \mathbb{F}_q^n.$$  (1)
This sequence of divisors can be constructed as follows. Let \( \alpha_1, \ldots, \alpha_g \in [0, 2g - 1] \cap H(P) \) where \( P \) is a place of degree one and \( H(P) \) is the Weiestrass semigroup at \( P \). Define \( \alpha_{n-g}, \ldots, \alpha_n \in [n, \infty) \) such that

\[
l(\alpha_i P) - l(\alpha_{i-1} P - D) \neq l(\alpha_{i-1} P - D) - l(\alpha_{i-1} P - D)
\]

and \( l(\alpha_n P) - l(\alpha_n P - D) = n \). Then the one-point codes from the sequence of divisors

\[
\alpha_1 P \leq \cdots \leq \alpha_g P \leq 2gP \leq (2g + 1)P \leq \cdots \leq (n - 1)P \leq \alpha_n P
\]

satisfy (1). Notice that \( C(D, \alpha_i P) \) is an \([n, i, d_i]_q\) code where \( \alpha_i = 2i - 2 \) for \( g + 1 \leq i \leq n - g - 1 \). If \( \deg(A_i) < n \), then

\[
n + 1 - i - g \leq d_i \leq n + 1 - i.
\]

Let \( G \) be a generator matrix of \( C(D, \alpha_n P) \) such that the matrix with rows \( \text{Row}_{n-i+1} G, \ldots, \text{Row}_n G \) is a generator matrix for \( C(D, A_i) \). This nested structure of codes allows us to bound the partial distances \( D_i \) of \( C(D, \alpha_n P) \) by

\[
D_i \geq d(C(D, \alpha_{n-i+1} P)) \geq n - \alpha_{n-i+1}.
\]

This bound combined with Lemma 1 yields the following result.

**Theorem 1** The exponent of the polar code with kernel \( G \) constructed using an AG code as above satisfies

\[
E(G) \geq \frac{1}{n} \left[ \log_n((n - g)! + \sum_{i=n-g+1}^n \log_n(d_i)) \right].
\]

Taking \( F \) to be a function field of genus \( g = 0 \), then one obtains the exponent of the RS kernel, which is also found in [5].

**Corollary 1** The exponent of a Reed-Solomon kernel \( G_{RS} \) over \( \mathbb{F}_q \) is

\[
E(G_{RS}) = \frac{\log_q(q!)}{q}.
\]

Another consequence of Theorem 1 is the asymptotic behavior of exponents of kernels constructed from codes over maximal function fields. Recall that a function field over \( \mathbb{F}_q \) of genus \( g \) is said to be maximal provided its number of places of degree one meets the Hasse-Weil bound; that is, the number of places of \( F \) of degree one is \( q + 1 + 2g\sqrt{q} \).

**Theorem 2** Let \( F/\mathbb{F}_q \) be a maximal function field of genus \( g \), and let \( G \) be a generator matrix of an AG code on \( F \) of length \( n = q + 2gq^{1/2} \) constructed as in (1). Then

\[
\lim_{q \to \infty} E(G) = 1.
\]

In the next subsection, we more closely examine kernels from codes over a particular maximal function field, the Hermitian function field.
3.1 Hermitian codes

Let \( F = F_{q^2}(x, y) \) be the function field of the curve

\[
y^q + y = x^{q+1}
\]

where \( q \) is a power of a prime; \( F \) is known as the Hermitian function field. The Hermitian function field over \( F_{q^2} \) has genus \( \frac{q(q-1)}{2} \) and \( q^3 + 1 \) places of degree one; hence, it is a maximal function field. A Hermitian one-point code is of the form \( C(D, aP_\infty) \), where \( D = \sum_{\alpha, \beta \in F_{q^2}, y = \alpha^{q+1}} P_{\alpha, \beta} \) and \( P_{\alpha, \beta} \) is a common zero of \( x - \alpha \) and \( y - \beta \). Mori and Tanaka considered generator matrices for Hermitian codes over fields of even characteristic, that is, over \( F_{2^m} \), as kernels of polar codes in [6]. More generally, applying Theorem 1 and the exact distances of one-point Hermitian codes [11], we have the result for any characteristic.

Corollary 2 The exponent of a Hermitian kernel \( G_H \) over \( F_{q^3} \) is bounded below by

\[
E(G) \geq \frac{1}{q^3} \log_{q^3} \left( (q^3 - q^2 + q)^{\frac{q-1}{2}} \prod_{j=1}^{q-1} \frac{(q^3 - (j-1)q)^{\frac{j}{2}}(q-1)^{\frac{j}{2}}(q^2 - jq)^{\frac{j}{2}}}{\prod_{i=1}^{j} (q^2 - jq - i)} \right),
\]

where \( a^! := \frac{a!}{(a-1)!} \).

Table 1 displays comparisons between the exponents of Reed-Solomon kernels and lower bounds on the exponents of Hermitian kernels. Note that the kernel size for the Reed-Solomon kernel over \( F_{q^2} \) is \( q^2 \times q^2 \), while the kernel size for the Hermitian kernel over \( F_{q^2} \) is \( q^3 \times q^3 \). The table suggests the next result, whose proof follows immediately from Theorem 1, Corollary 1, and Corollary 2.

Proposition 1 Let \( G_H \) be a Hermitian kernel over \( F_{q^3} \), and let \( G_{RS} \) be a Reed-Solomon kernel also over \( F_{q^2} \). Then

\[
E(G_{RS}) \leq E(G_H)
\]

for \( q \geq 3 \).
3.2 Suzuki codes

In this subsection, we see that the asymptotic behavior of the exponent in Theorem 2 is not restricted to maximal function fields. To do so, we investigate codes from the Suzuki function field, a function field which is not maximal yet has a large number of places of degree one for its genus.

Let \( F = \mathbb{F}_q(x,y) \) be the function field of the Suzuki curve

\[
y^{2^{2n+1}} - 2^n y = x^{2^{2n+1}} - x
\]

where \( q = 2^{2n+1} \) and \( n \) is a positive integer. Then the genus of \( F \) is \( g = \sqrt{q^2/(q-1)} \), and \( F \) has exactly \( q^2 + 1 \) places of degree one. The Suzuki one-point code is of the form \( C(D, aP_\infty) \), where \( D = \sum_{\alpha, \beta \in \mathbb{F}_{2^{2n+1}}} P_{\alpha, \beta} \) and \( P_{\alpha, \beta} \) is a common zero of \( x - \alpha \) and \( y - \beta \). Theorem 1 then yields the following result.

**Corollary 3** Let \( G_{Suz} \) be a Suzuki kernel over \( \mathbb{F}_q \) where \( q = 2^{2n+1} \). Then

\[
E(G_{Suz}) \geq \frac{1}{q^2} \log_{q^2} \left( \left( q^2 - \sqrt{q^2/(q-1)} \right)! \right).
\]

The exact minimum distance of Suzuki one-point codes over \( \mathbb{F}_8 \) are known \cite{2}. We further explore this function field in the example below.

**Example 1** Let \( F = \mathbb{F}_8(x,y) \) be function field of the Suzuki curve \( y^8 - y = x^4(x^8 - x) \). The Suzuki one-point code over this function field is of length 64, and the Suzuki kernel \( G_{Suz}(8) \) over \( \mathbb{F}_8 \), where \( \alpha \) is a primitive element of \( \mathbb{F}_8 \) is defined by basis functions

\[
f_1 = X, \quad f_2 = Y, \quad f_3 = X^5 + Y^4, \quad f_4 = XY^4 + X^{20} + Y^{16}
\]

and is shown below.

\[
G_{Suz}(8) = \begin{pmatrix}
(0,0) & (0,1) & \cdots & (0,\alpha) & \cdots & (\alpha^4, \alpha^5) & (\alpha^6, \alpha^6) \\
0 & 1 & \cdots & 1 & \cdots & (1 + \alpha^6)(1 + \alpha^8 + \alpha^{10}) & 0 \\
1 & 1 & \cdots & 1 & \cdots & \alpha^6 & 1 \\
0 & 0 & \cdots & 0 & \cdots & \alpha^8 & 1 \\
1 & 1 & \cdots & 1 & \cdots & \alpha^8 & 1
\end{pmatrix}
\]

The bounds on the partial distances of this matrix are bounded by the exact minimum distances of the one-point Suzuki codes

\[
C(D, P_\infty) \subseteq C(D, 8P_\infty) \subseteq \cdots \subseteq C(D, 77P_\infty),
\]

which according to \cite{2} are

\[
\{64, 56, \ldots, 8\}.
\]

Hence, Theorem 1 implies \( E(G_{Suz}(8)) \geq 0.65555 \).

Table 2 compares the exponents of Reed-Solomon kernels and lower bounds on the exponents of Suzuki kernels. As with Hermitian kernels, Suzuki kernels yield larger exponents than Reed-Solomon kernels; however, the larger exponent comes at the price of a larger kernel size.
<table>
<thead>
<tr>
<th>Kernel</th>
<th>Exponent</th>
<th>Size of Kernel</th>
<th>Exponent</th>
<th>Size of Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reed-Solomon</td>
<td>0.63747</td>
<td>8 × 8</td>
<td>0.73540</td>
<td>32 × 32</td>
</tr>
<tr>
<td>Suzuki</td>
<td>0.65555</td>
<td>64 × 64</td>
<td>0.73635</td>
<td>1024 × 1024</td>
</tr>
</tbody>
</table>

Table 2 Lower bounds on exponents of Reed-Solomon and Suzuki kernels over $\mathbb{F}_q$ where $q = 2^{2n+1}$

**Proposition 2** Let $q = 2^{2n+1}$. Let $G_{Suz}$ be a Suzuki kernel over $\mathbb{F}_q$, and let $G_{RS}$ be a Reed-Solomon kernel also over $\mathbb{F}_q$. Then

$$E(G_{RS}) \leq E(G_{Suz})$$

for all $q = 2^{2n+1}$ where $n \geq 1$.

The limiting behavior of the exponent in Theorem 2 is not restricted to maximal function fields. In fact, kernels from Suzuki one-point codes display similar asymptotics.

**Theorem 3** Let $G_{Suz}$ be a Suzuki kernel over $\mathbb{F}_q$ where $q = 2^{2n+1}$. Then

$$\lim_{q \to \infty} E(G_{Suz}) = 1.$$ 

3.3 Shortening an AG kernel is a special case of a multipoint AG code

The method of shortening can be used to create smaller kernels with large exponent. In [4], Korada, Şaşoğlu, and Urbanke used repeated shortening of a BCH code to create the smallest binary kernel with exponent exceeding $\frac{1}{2}$, that of Arikan’s original kernel $G_2$. To shorten a kernel matrix, first find the column $j$ with the longest run of zeros at the end of the column. Then find the row $i$ with the last nonzero element of column $j$. Add row $i$ to all the rows with a nonzero element in column $j$. Finally, remove column $j$ and row $i$. As the next result shows, shortening applied to AG code kernels is a special case of a multipoint code construction.

**Theorem 4** Let $\alpha_1 \leq \ldots \leq \alpha_n$ be integers such that

$$C(D, \alpha_1 P) \subset \ldots \subset C(D, \alpha_n P) = \mathbb{F}_q^n.$$ 

Let $G$ be the generator matrix of $C(D, \alpha_n P)$. Suppose the method of shortening is applied to the $j$th column of $G$. Then

$$C(D - P_j, \alpha_1 P - P_j) \subset \ldots \subset C(D - P_j, \alpha_n P - P_j) = \mathbb{F}_q^{n-1},$$

and $G$ now corresponds to the generator matrix of $C(D - P_j, \alpha_n P - P_j)$, which is a two-point code.

Note that we can apply this method repeatedly, which will result in other multipoint codes.
Example 2 Consider the Hermitian code, \( C(D, 8P_\infty) \), over \( \mathbb{F}_4 \). Then the Hermitian kernel is the matrix \( G_H(4) \) below:

\[
G_H(4) = \begin{pmatrix}
(0,0) & (0,1) & (1, \alpha) & (1, \alpha^2) & (\alpha, \alpha) & (\alpha, \alpha^2) & (\alpha^2, \alpha) & (\alpha^2, \alpha^2) \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
X & 0 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 \\
Y & 0 & 1 & \alpha & \alpha^2 & \alpha & \alpha^2 & \alpha & \alpha \\
X^2 & 0 & 0 & 1 & 1 & \alpha^2 & \alpha^2 & 1 & 1 & \alpha \\
XY & 0 & 0 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 & 1 \\
X^3 & 0 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 \\
X^2Y & 0 & 0 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 & 1 \\
X^4 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 \\
\end{pmatrix}
\]

The columns are indexed by \((\alpha, \beta)\) such that \( P_{\alpha,\beta} \) is a place of degree one of the Hermitian function field over \( \mathbb{F}_4 \), and the rows are indexed by functions in a basis of the Riemann-Roch space \( \mathcal{L}(8P_\infty) \). We then pick the column with the longest run of zeros on the bottom, which is column 1. Since row 1 is the last row with a 1 in column 1, we will remove column 1 and row 1. The resulting kernel is:

\[
(0,1) & (1, \alpha) & (1, \alpha^2) & (\alpha, \alpha) & (\alpha, \alpha^2) & (\alpha^2, \alpha) & (\alpha^2, \alpha^2) \\
X & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 \\
Y & 1 & \alpha & \alpha^2 & \alpha & \alpha^2 & \alpha & \alpha \\
X^2 & 0 & 1 & 1 & \alpha^2 & \alpha^2 & \alpha & \alpha \\
XY & 0 & \alpha & \alpha^2 & \alpha^2 & 1 & 1 & \alpha \\
X^3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
X^2Y & 0 & \alpha & \alpha^2 & 1 & \alpha & \alpha^2 & 1 \\
X^4 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 & \alpha^2 & \alpha^2 \\
\end{pmatrix}
\]

which may be obtained from a generator matrix of the two-point Hermitian code, \( C(D - P_{(0,0)}, 8P_\infty - P_{(0,0)}) \).

4 Probability of block error

We can also consider the probability of block error using polar coding. Let \( W \) be a \( q \)-ary DMC. If \( G \) is a matrix that polarizes according to [5], then the exponent helps bound the block error probability under successive cancellation decoding. Let \( P_e \) be the best achievable probability of block error under SC decoding for polar coding over \( W \) using kernel \( G \). Using techniques similar to [1] and [7], the following result holds.

Theorem 5 Consider polar coding over a \( q \)-ary DMC using kernel \( G \) at a fixed rate \( 0 < R < I(W) \) with block length \( N = \ln a \). Then

\[
P_e = O(2^{-\ln \beta})
\]

for \( 0 < \beta < E(G) \).
5 Conclusions

AG codes can be used to create kernels which give better exponents for polar codes. Certain AG codes have known Riemann-Roch bases, so we have explicit kernels that may be described simply by basis functions and places. However, these better exponents come at a cost of a larger kernel. Shortening may be used to create smaller kernels, but this might decrease the exponent; multipoint AG code kernels may do the same. A hybrid approach might allow one to balance these competing goals.

References

New Inequalities for $q$-ary Constant-Weight Codes

Hyun Kwang Kim and Phan Thanh Toan
Department of Mathematics
Pohang University of Science and Technology
Pohang 790-784, Republic of Korea
{hkkim,pttoan}@postech.ac.kr

Abstract. Using double counting, we prove Delsarte inequalities for $q$-ary codes and their improvements. Applying the same technique to $q$-ary constant-weight codes, we obtain new inequalities for $q$-ary constant-weight codes.

Keywords: binary code, Delsarte inequality, linear programming, $q$-ary code.

1 Introduction

Let $\mathbb{F}_q$ be a finite field with $q$ elements and let $n$ be a positive integer. The (Hamming) distance between two vectors $u$ and $v$ in $\mathbb{F}_q^n$, denoted by $d(u,v)$, is the number of coordinates where they differ. A $q$-ary code of length $n$ is a subset of $\mathbb{F}_q^n$. If $C$ is a $q$-ary code (of length $n$), then each element of $C$ is called a codeword. The size of $C$, denoted by $|C|$, is the number of codewords in $C$. Let $C$ be a $q$-ary code of length $n$. The distance distribution \{$B_i\}_{i=0}^n$ of $C$ is defined by

$$B_i = \frac{1}{|C|} \cdot |\{(u,v) \mid u,v \in C, d(u,v) = i\}|.$$  \hspace{1cm} (1)

It is well known that for each $k = 1, 2, \ldots, n$,

$$\sum_{i=0}^{n} P_k(n;i)B_i \geq 0, \hspace{1cm} (2)$$

where $P_k(n;x)$ is the Krawtchouk polynomial given by

$$P_k(n;x) = \sum_{j=0}^{k} (-1)^j (q-1)^{k-j} \binom{x}{j} \binom{n-x}{k-j}. \hspace{1cm} (3)$$

The inequalities (2) are called the Delsarte inequalities, which were proved by Delsarte in 1973 [1]. These inequalities are extremely useful in coding theory (they are used to give upper bounds on sizes of $q$-ary codes through linear programming [1, 2]).
For \( q = 2 \), using double counting, Kang, Kim, and Toan were able to prove simultaneously the Delsarte inequalities and two known improvements when the size of the binary code is odd and congruent to 2 modulo 4, respectively [3]. Applying the same technique to binary constant-weight codes, they obtained new linear inequalities, which allowed them to give new upper bounds on sizes of binary constant-weight codes.

The purpose of this paper is to generalize the results in [3] to arbitrary \( q \)-ary codes. In Section 2, we prove simultaneously Delsarte inequalities for \( q \)-ary codes and their known improvements (Theorem 1). Applying the same technique to \( q \)-ary constant-weight codes, in Section 3, we obtain new inequalities for \( q \)-ary constant-weight codes (Theorem 2). These inequalities generalize inequalities shown by Östergård and Svanström in [4]. As in the binary case (see [3]), these inequalities can be used to get new upper bounds on sizes of \( q \)-ary constant-weight codes.

## 2 Delsarte Inequalities for \( q \)-ary Codes

In this section, we give a short proof of the Delsarte inequalities for \( q \)-ary codes and their known improvements. For the improvements of the Delsarte inequalities, see [5, Theorem 4]. Our proof is a generalization of the proof appearing in [3] for the binary case \((q = 2)\). As in [3], when applying the same technique in our proof to \( q \)-ary constant-weight codes, we obtain new inequalities for \( q \)-ary constant-weight codes in Section 3, which can be used to get new upper bound on sizes of \( q \)-ary constant-weight codes. We note that a simple proof of the Delsarte inequalities appeared in [6].

For two vectors \( a = (a_1, a_2, \ldots, a_j) \) and \( b = (b_1, b_2, \ldots, b_j) \) in \( \mathbb{F}_q^j \), the inner product of \( a \) and \( b \) is defined by \( a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_j b_j \). Let \( \mathbb{F}_q^* \) be the set of nonzero elements in \( \mathbb{F}_q \). For each \( a = (a_1, a_2, \ldots, a_j) \in (\mathbb{F}_q^*)^j \), denote

\[
N(a) = |\{b = (b_1, b_2, \ldots, b_j) \in (\mathbb{F}_q^*)^j \mid a \cdot b \neq 0\}| \tag{4}
\]

and

\[
Z(a) = |\{b = (b_1, b_2, \ldots, b_j) \in (\mathbb{F}_q^*)^j \mid a \cdot b = 0\}|. \tag{5}
\]

**Proposition 1.** For each \( a = (a_1, a_2, \ldots, a_j) \in (\mathbb{F}_q^*)^j \),

\[
N(a) = q - \frac{1}{q}[(q - 1)^j - (-1)^j] \tag{6}
\]

and

\[
Z(a) = (q - 1)^j - \frac{1}{q}[(q - 1)^j - (-1)^j]. \tag{7}
\]

**Proof.** The proof is straightforward by using induction on \( j \).
Lemma 1. Suppose that $n_1 + n_2 + \cdots + n_h = M$, where $M$ is a constant and $n_c \ (c = 1, 2, \ldots, h)$ are nonnegative integers. Then the sum

$$\sum_{c<i} n_c n_d$$

is maximum if and only if $|n_c - n_d| \leq 1$ for all $c \neq d$.

Let $C$ be a $q$-ary code of length $n$ with distance distribution $\{B_i\}_{i=0}^n$ and let $M = |C|$. Consider $C$ as an $M \times n$ matrix (where each $c \in C$ is a row). The $m$th row of $C$ is denoted by $(c_{m1}, c_{m2}, \ldots, c_{mn})$, $m = 1, 2, \ldots, M$. Let $u_1', u_2', \ldots, u_n'$ be the $n$ columns of $C$. Each $u_i'$ can be considered as a vector in $\mathbb{F}_q$. Write $\mathbb{F}_q = \{0, 1\}$. For each $c = 1, 2, \ldots, q$ and each vector $a = (a_1, a_2, \ldots, a_M) \in \mathbb{F}_q^M$, denote

$$x_c(a) = \{|j | a_j = \omega_c\}.$$  

By definition, $\sum_{c=1}^q x_c(a) = M$. For each $k = 1, 2, \ldots, n$, let

$$S(k) = \sum_{\alpha \in (\mathbb{F}_q)^k} \sum_{i_1 < \cdots < i_k} x_{\alpha_1 u'_1 + \cdots + \alpha_k u'_k}.$$  

Lemma 2. Suppose that $C$ is a $q$-ary code of length $n$ with size $M$. Then for each $k = 1, 2, \ldots, n$,

$$S(k) \leq (q - 1)^k \binom{n}{k} \left[ \frac{q - 1}{2q} M^2 + \frac{r(q - 1)}{2q} \right],$$  

where $r$ is the remainder when dividing $M$ by $q$.

For each $k = 1, 2, \ldots, n$, we introduce the polynomials

$$P_k^-(n; x) = \frac{1}{2} \sum_{j=0}^{k} [(q - 1)^j - (-1)^j](q - 1)^{k-j} \binom{n-x}{j} \binom{x}{k-j}$$

and

$$P_k^+(n; x) = (q - 1)^k \binom{n}{k} - P_k^-(n; x).$$

Remark 1. It follows from definition that $P_k^+(n; x) + P_k^-(n; x) = (q - 1)^k \binom{n}{k}$ and $P_k^+(n; x) - P_k^-(n; x) = \sum_{j=0}^{k} (-1)^j(q - 1)^{k-j} \binom{n-x}{j} \binom{x}{k-j}$ = $P_k(n; x)$ is the Krawtchouk polynomial.

Lemma 3. Suppose that $C$ is a $q$-ary code of length $n$ with size $M$ and distance distribution $\{B_i\}_{i=0}^n$. Then for each $k = 1, 2, \ldots, n$,

$$\sum_{i=1}^n P_k^+(n; i)B_i = \frac{q}{(q - 1)M} S(k)$$

and
\[
\sum_{i=1}^{n} P^{+}_k (n; i) B_i = (M - 1)(q - 1)^k \binom{n}{k} - \frac{q}{(q - 1)M} S(k).
\]
(15)

Proof. Write \( C = (c_{mi}) \). Let \( S_1(k) \) be the number of pairs \((A, \alpha)\) satisfying the following conditions.

(i) \( A \) is a \( 2 \times k \) matrix

\[
A = \begin{pmatrix}
c_{mi_1} & c_{mi_2} & \cdots & c_{mi_k}
\end{pmatrix}
\]

such that \( m \neq l \) and \( i_1 < i_2 < \cdots < i_k \).

(ii) \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \) is an element in \((\mathbb{F}_q)^k\) such that

\[
\alpha_1(c_{mi_1} - c_{li_1}) + \cdots + \alpha_k(c_{mi_k} - c_{li_k}) \neq 0.
\]
(17)

We count in two ways the number of pairs \((A, \alpha)\) satisfying the above conditions. For two rows \( u = (c_{m1} \ c_{m2} \cdots \ c_{mn}) \) and \( v = (c_{11} \ c_{12} \cdots \ c_{1n}) \), we choose a set \( I_1 \) containing \( j \) coordinates \((0 \leq j \leq d(u,v))\) where \( u \) and \( v \) differ and choose another set \( I_2 \) containing \( k - j \) coordinates where \( u \) and \( v \) are the same. Let \( \{i_1, i_2, \ldots, i_k\} \) be such that \( \{i_1, i_2, \ldots, i_k\} = I_1 \cup I_2 \) and \( i_1 < i_2 < \cdots < i_k \).

For each \( j \), there are exactly \( \binom{d(u,v)}{j} \left( n - d(u,v) \right)^{k-j} \) choices for such \( i_1 < i_2 < \cdots < i_k \). Now fix \( u, v, j \), and \( i_1 < i_2 < \cdots < i_k \). We count the number of \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in (\mathbb{F}_q)^k \) such that \( \alpha_1(c_{mi_1} - c_{li_1}) + \cdots + \alpha_k(c_{mi_k} - c_{li_k}) \neq 0 \) or

\[
\sum_{t \in I_1} \alpha_t(c_{mi_t} - c_{li_t}) \neq 0.
\]
(18)

By Proposition 1, there are exactly \( \frac{q - 1}{q}[(q - 1)^j - (-1)^j] \) choices for \((\alpha_t)_{t \in I_1}\) such that (18) holds. Since \((\alpha_t)_{t \in I_2}\) can be chosen arbitrarily, we get in total

\[
\frac{q - 1}{q} [(q - 1)^j - (-1)^j] \cdot (q - 1)^{k-j}
\]
(19)

choices for \( \alpha \). In conclusion,

\[
S_1(k) = \sum_{u, v \in C \atop u \neq v} \sum_{j=0}^{d(u,v)} \frac{q - 1}{q}[(q - 1)^j - (-1)^j](q - 1)^{k-j} \binom{n - d(u,v)}{j} \binom{n - d(u,v)}{k - j}
\]

\[
= \sum_{u, v \in C \atop u \neq v} \frac{2(q - 1)}{q} P^{+}_k (n; d(u,v))
\]

\[
= \sum_{i=1}^{n} \frac{q}{q} \sum_{u, v \in C \atop d(u,v) = i} n \sum_{u, v \in C \atop d(u,v) = i} P^{+}_k (n; i)
\]

\[
= \frac{2(q - 1)M}{q} \sum_{i=1}^{n} P^{+}_k (n; i) B_i.
\]
(20)
Now for each $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in (\mathbb{F}_q^*)^k$ and $k$ columns $u'_{i_1}, u'_{i_2}, \ldots, u'_{i_k}$ ($i_1 < i_2 < \cdots < i_k$), there are
\[ \sum_{c<d} 2 x_c (\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \cdot x_d (\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \tag{21} \]
choices for $m \neq l$ such that $\alpha_1 c_{m i_1} + \cdots + \alpha_k c_{m i_k} \neq \alpha_1 c_{l i_1} + \cdots + \alpha_k c_{l i_k}$. It follows that
\[ S_1(k) = \sum_{\alpha \in (\mathbb{F}_q^*)^k} \sum_{i_1 < \cdots < i_k} 2 x_c (\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \cdot x_d (\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \]
when dividing $M$ by $q$, then for each $k = 1, 2, \ldots, n$,
\[ S_1(k) = 2 S(k). \tag{22} \]
(20) and (22) give
\[ \sum_{i=1}^{n} P_k^{-}(n; i) B_i = \frac{q}{(q-1)M} S(k). \tag{23} \]
Finally,
\[ \sum_{i=1}^{n} P_k^{+}(n; i) B_i = \sum_{i=1}^{n} \left[ (q-1)^k \binom{n}{k} - P_k^{-}(n; i) \right] B_i \]
\[ = \sum_{i=1}^{n} (q-1)^k \binom{n}{k} B_i - \sum_{i=1}^{n} P_k^{+}(n; i) B_i \]
\[ = (M-1)(q-1)^k \binom{n}{k} - \frac{q}{(q-1)M} S(k). \tag{24} \]

**Theorem 1.** (Improved Delsarte inequalities). Suppose that $C$ is a $q$-ary code of length $n$ with size $M$ and distance distribution $\{B_i\}_{i=0}^{n}$. If $r$ is the remainder when dividing $M$ by $q$, then for each $k = 1, 2, \ldots, n$,
\[ \sum_{i=0}^{n} P_k(n; i) B_i \geq \frac{1}{M} r(q-r)(q-1)^{k-1} \binom{n}{k}. \tag{25} \]

**Proof.** By Lemmas 3 and 2,
\[ - \sum_{i=1}^{n} P_k(n; i) B_i = \sum_{i=1}^{n} P_k^{-}(n; i) B_i - \sum_{i=1}^{n} P_k^{+}(n; i) B_i \]
\[ = -(M-1)(q-1)^k \binom{n}{k} + \frac{2q}{(q-1)M} S(k) \]
\[ \leq -(M-1)(q-1)^k \binom{n}{k} \]
\[ + \frac{2q}{(q-1)M} (q-1)^k \binom{n}{k} \left[ \frac{q-1}{2q} M^2 + \frac{r(r-q)}{2q} \right] \]
\[ = (q-1)^k \binom{n}{k} - \frac{1}{M} r(q-r)(q-1)^{k-1} \binom{n}{k}. \tag{26} \]
Since $P_k(n; 0) = (q-1)^k \binom{n}{k}$ and $B_0 = 1$, the above inequality gives the desired result.
3 New Inequalities for $q$-ary Constant-Weight Codes

Let $C$ be a $q$-ary code of length $n$ and constant-weight $w$. Let $M = |C|$. As before, consider $C$ as an $M \times n$ matrix (where each $c \in C$ is a row). The $m$th row of $C$ is denoted by $(c_m1, c_m2, \ldots, c_mn)$, $m = 1, 2, \ldots, M$. Let $u'_1, u'_2, \ldots, u'_n$ be the $n$ columns of $C$. Write $F_q = \{0 = \omega_1, \omega_2, \ldots, \omega_q\}$.

Lemma 4. For each $k = 1, 2, \ldots, n$,
\[
\sum_{\alpha \in (F_q^*)^k} \sum_{i_1 < \cdots < i_k} x_c (\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = \frac{2(q-1)}{q} MP_k^{-}(n; w). \tag{27}
\]

Lemma 5. Suppose that $A = (n_{ci})$ is a $q \times N$ matrix with nonnegative entries such that $\sum_{i=1}^{N} \sum_{c=1}^{q} n_{ci} = M'$ and $\sum_{c=1}^{q} n_{ci} = M$ for all $i = 1, 2, \ldots, N$, where $M'$ and $M$ are constants. Then the sum
\[
\sum_{i=1}^{N} \sum_{c<d} n_{ci} n_{di} \tag{28}
\]
is maximum if and only if $|n_{1i} - n_{1j}| \leq 1$ for all $i \neq j$ and $|n_{ci} - n_{di}| \leq 1$ for all $i = 1, 2, \ldots, N$ and all $2 \leq c < d \leq q$.

Let $C$ be a $q$-ary code of length $n$ and constant-weight $w$. Let $M = |C|$. For each $k = 1, 2, \ldots, n$, denote
\[
T_1(k) = \left[(q-1)^k \binom{n}{k} - r_k\right] (M - q_k) q_k + r_k (M - q_k - 1)(q_k + 1), \tag{29}
\]
\[
T_2(k) = \left[(q-1)^k \binom{n}{k} - r_k\right] \left[\frac{(q-1-t_k)}{2} s_k^2 \right. \\
\left. + (q-1-t_k) t_k s_k (s_k + 1) + \left(\frac{t_k}{2}\right) (s_k + 1)^2 \right], \tag{30}
\]
and
\[
T_3(k) = r_k \left[\frac{(q-1-t'_k)}{2} s'_{k}^2 \right. + (q-1-t'_k) t'_k s'_k (s'_k + 1) \\
\left. + \left(\frac{t'_k}{2}\right) (s'_k + 1)^2 \right]. \tag{31}
\]

where
\begin{itemize}
  \item $q_k$ and $r_k$ are the quotient and the remainder, respectively, when dividing $\frac{2(q-1)M P_k^{-}(n; w)}{q}$ by $(q-1)^k \binom{n}{k}$,
  \item $s_k$ and $t_k$ are the quotient and the remainder, respectively, when dividing $q_k$ by $(q-1)$,
\end{itemize}
New Inequalities for q-ary Constant-Weight Codes

Lemma 6. Suppose that C is a q-ary code of length n and constant-weight w. Let S(k) be defined by (10). Then for each k = 1, 2, ..., n,

\[ S(k) \leq T_1(k) + T_2(k) + T_3(k). \]  

(32)

Proof. Let \( N = (q - 1)^k \binom{n}{k} \) and let A be the \( q \times N \) matrix defined as follows.

\( \circ \) The rows of A are indexed by \( c \) (\( c = 1, 2, ..., q \)).

\( \circ \) The columns of A are indexed by pairs \( (\alpha, i_1 < \cdots < i_k) \), where \( \alpha \in (\mathbb{F}_q^*)^k \) and \( 1 \leq i_1 < \cdots < i_k \leq q \).

\( \circ \) The \( \left( c, (\alpha, i_1 < \cdots < i_k) \right) \)th entry of A is \( x_c(\alpha u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \).

Recall that \( S(k) = \sum_{\alpha \in (\mathbb{F}_q^*)^k} \sum_{i_1 < \cdots < i_k} x_c(\alpha u'_{i_1} + \cdots + \alpha_k u'_{i_k}) \cdot x_d(\alpha u'_{i_1} + \cdots + \alpha_k u'_{i_k}). \)  

(33)

By Lemma 4,

\[ \sum_{\alpha \in (\mathbb{F}_q^*)^k} \sum_{\substack{2 \leq c \leq q \\alpha < \cdots < \alpha_k}} x_c(\alpha u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = \frac{2(q - 1)}{q} MP_k^{-}(n; w). \]  

(34)

Also, by definition,

\[ \sum_{1 \leq c \leq q} x_c(\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = M \]  

(35)

Hence, we can apply Lemma 5 to the matrix A. Lemma 5 says that \( S(k) \) is maximum when

\[ \sum_{2 \leq c \leq q} x_c(\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = q_k \]  

(36)

or \( x_1(\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = M - q_k \)  

(37)

for \( (q - 1)^k \binom{n}{k} - r_k \) pairs \( (\alpha, i_1 < \cdots < i_k) \) and

\[ \sum_{2 \leq c \leq q} x_c(\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = q_k + 1 \]  

(38)

or \( x_1(\alpha_1 u'_{i_1} + \cdots + \alpha_k u'_{i_k}) = M - q_k - 1 \)  

(39)

for the other \( r_k \) pairs \( (\alpha, i_1 < \cdots < i_k) \). Also, for each pair \( (\alpha, i_1 < \cdots < i_k) \), the following must hold.

\( \circ \) \( s_k' \) and \( t_k' \) are the quotient and the remainder, respectively, when dividing \( q_k + 1 \) by \( (q - 1) \).
If $\sum_{2\leq c\leq q} x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = q_k$, then
\[ x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = s_k \]  
(40)
for $q - 1 - t_k$ values of $c \geq 2$ and
\[ x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = s_k + 1 \]  
(41)
for the other $t_k$ values of $c \geq 2$.

If $\sum_{2\leq c\leq q} x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = q_k + 1$, then
\[ x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = s_k' \]  
(42)
for $q - 1 - t_k'$ values of $c \geq 2$ and
\[ x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') = s_k' + 1 \]  
(43)
for the other $t_k'$ values of $c \geq 2$.

Therefore,
\[
S(k) = \sum_{\alpha \in (\mathbb{F}_q^*)^k} \sum_{2 \leq c \leq q} x_1(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') \cdot x_d(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') \\
+ \sum_{\alpha \in (\mathbb{F}_q^*)^k} \sum_{2 \leq c \leq q} x_c(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') \cdot x_d(\alpha_1 u_{i_1}' + \cdots + \alpha_k u_{i_k}') \\
\leq \left[ (q - 1)^k {n \choose k} - r_k \right] (M - q_k)q_k + r_k(M - q_k - 1)(q_k + 1) \\
+ \left[ (q - 1)^k {n \choose k} - r_k \right] \left[ \left( \frac{q - 1 - t_k}{2} \right) s_k^2 + (q - 1 - t_k) t_k s_k (s_k + 1) + \left( \frac{t_k}{2} \right) (s_k + 1)^2 \right] \\
+ r_k \left[ \left( \frac{q - 1 - t_k'}{2} \right) s_k'^2 + (q - 1 - t_k') t_k' s_k' (s_k' + 1) + \left( \frac{t_k'}{2} \right) (s_k' + 1)^2 \right] \\
= T_1(k) + T_2(k) + T_3(k). \]  
(44)

**Theorem 2.** Suppose that $\{B_i\}_{i=0}^n$ is the distance distribution of a $q$-ary code $C$ of length $n$ and constant-weight $w$. Then for each $k = 1, 2, \ldots, n$,
\[
\sum_{i=1}^n P_k(n; i) B_i \geq (M - 1)(q - 1)^k {n \choose k} - \frac{2q}{(q - 1)M} T(k), \]  
(45)
where $T(k) = T_1(k) + T_2(k) + T_3(k)$. 
Proof. Considering \( C \) as a \( q \)-ary code and applying Lemma 3, we get

\[
- \sum_{i=1}^{n} P_k(n; i)B_i = \sum_{i=1}^{n} P^-_k(n; i)B_i - \sum_{i=1}^{n} P^+_k(n; i)B_i = -(M-1)(q-1)^k \binom{n}{k} + \frac{2q}{(q-1)M} S(k). \tag{46}
\]

By Lemma 6,

\[
S(k) \leq T_1(k) + T_2(k) + T_3(k) = T(k). \tag{47}
\]

Hence,

\[
- \sum_{i=1}^{n} P_k(n; i)B_i \leq -(M-1)(q-1)^k \binom{n}{k} + \frac{2q}{(q-1)M} T(k).
\]

When \( k = 1 \), Theorem 2 implies the following corollary, which appeared in [4] (see [4, Theorem 12]).

**Corollary 1.** If there exists a \( q \)-ary code of length \( n \), constant-weight \( w \), and minimum distance \( \geq d \), then

\[
M(M-1)d \leq 2t \sum_{i=0}^{q-2} \sum_{j=i+1}^{q-1} M_iM_j + 2(n-t) \sum_{i=0}^{q-2} \sum_{j=i+1}^{q-1} M'_iM'_j, \tag{48}
\]

where

- \( k \) and \( t \) are the quotient and the remainder, respectively, when dividing \( Mw \) by \( n \),
- \( M_0 = M - k - 1 \), \( M'_0 = M - k \), \( M_i = \lfloor (k+i)/(q-1) \rfloor \), and \( M'_i = \lfloor (k+i-1)/(q-1) \rfloor \).

Theorem 2 can be used to get upper bounds on sizes of \( q \)-ary constant-weight codes. For \( q = 2 \), new upper bounds on sizes of binary constant-weight codes obtained from Theorem 2 are given in [3]. Corollary 1 was used in [4] to give upper bounds (sometimes exact values) on sizes of ternary constant-weight codes. For \( q \geq 3 \), the following example shows that new upper bounds on sizes of \( q \)-ary constant-weight codes can also be obtained from Theorem 2.

**Example 1.** Let \( A_q(n, d, w) \) denote the largest possible size of a \( q \)-ary code of length \( n \), constant-weight \( w \), and minimum distance at least \( d \). Suppose that \( q = 3 \) and \( (n, d, w) = (9, 3, 7) \). The best known upper bound for \( A_3(9, 3, 7) \) is \( A_3(9, 3, 7) \leq 576 \) (see [4, Table 5]). Suppose that \( A_3(9, 3, 7) = 576 \). Let \( C \) be a code whose size attains the upper bound and let \( \{B_i\}_{i=0}^{n} \) be the distance distribution of \( C \). Then Theorem 2 gives the following inequalities (note that
\( B_1 = B_2 = 0 \) since \( C \) has minimum distance \( \geq 3 \).

\[
\begin{align*}
9B_3 + 6B_4 + 3B_5 - 3B_7 - 6B_8 - 9B_9 & \geq 270 \\
27B_3 + 6B_4 - 6B_5 - 9B_6 - 3B_7 + 12B_8 + 36B_9 & \geq -108 \\
15B_3 - 24B_4 + 18B_5 + 6B_6 + 21B_7 - 84B_9 & \geq -294 \\
-72B_3 - 39B_4 + 21B_5 + 27B_6 - 21B_7 - 42B_8 + 126B_9 & \geq -1890 \\
-108B_3 + 42B_4 + 39B_5 - 36B_6 - 21B_7 + 84B_8 - 126B_9 & \geq -3969 \\
48B_3 + 72B_4 - 48B_5 - 15B_6 + 63B_7 - 84B_8 + 84B_9 & \geq -4942 \\
144B_3 - 48B_4 - 24B_5 + 54B_6 - 57B_7 + 48B_8 - 36B_9 & \geq -4194 \\
-48B_4 + 48B_5 - 36B_6 + 24B_7 - 15B_8 + 9B_9 & \geq -2160 \\
-64B_3 + 32B_4 - 16B_5 + 8B_6 - 4B_7 + 2B_8 - B_9 & \geq -1480 / 3 \\
\end{align*}
\]

Since \( B_0 = 1 \) and \( B_1 = B_2 = 0 \), 

\[
1 + \sum_{i=3}^{9} B_i = \sum_{i=0}^{9} B_i = |C| = 576. 
\]

Consider the following linear programming (where the \( B_i \) are considered as variables)

\[
\max \left( 1 + \sum_{i=3}^{9} B_i \right) 
\]

subject to \( B_i \geq 0 \), \( i = 3, 4, \ldots, 9 \) and subject to the above inequalities. Solving this linear programming, we get the maximum value of \( 1 + \sum_{i=3}^{9} B_i \) is 12094/21, which is less than 576. This contradiction shows that such a code \( C \) does not exist. Therefore,

\[
A_3(9,3,7) \leq 575. 
\]

References

Towards the Optimality of Feistel Ciphers with Substitution-Permutation Functions

Extended Abstract

Kyoji Shibutani · Andrey Bogdanov

Abstract We explore the optimality of balanced Feistel ciphers with SP-type F-functions with respect to their resistance against differential and linear cryptanalysis. Instantiations of Feistel ciphers with the wide class of \((\text{SP})^u\) and \((\text{SP})^u S\) F-functions are considered: One F-function can contain an arbitrary number of S-box layers interleaved with linear diffusion. For the matrices with maximum diffusion, it is proven that SPS and SPSP F-functions are optimal in terms of the proportion of active S-boxes in all S-boxes – a common efficiency metric for substitution-permutation ciphers. Interestingly, one SP-layer in the F-function is not enough to attain optimality whereas taking more than two S-box layers does not increase the efficiency either.

Keywords block cipher · balanced Feistel networks · differential cryptanalysis · linear cryptanalysis · active S-boxes · MDS codes

Mathematics Subject Classification (2000) 94A60

1 Introduction

Balanced Feistel networks (BFNs) are one of the most widely used structures for a block cipher. In fact, BFNs are adopted in a large number of symmetric key primitives, e.g., the former U.S. encryption standard DES [13], the current Russian encryption standard GOST blockcipher [15], and KASUMI which is the core of A5/3 cryptosystem in mobile networks [19]. Besides, a considerable number of analytic papers for the structure of the BFNs and the specific instantiation of the BFNs have been published since it was developed in the 1970s. However, the optimal design strategy with respect to both the security and the efficiency for its F-function is still an open problem.

K. Shibutani
Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan
E-mail: Kyoji.Shibutani@jp.sony.com

A. Bogdanov
Technical University of Denmark
E-mail: a.bogdanov@mat.dtu.dk
This article addresses this problem in a wide class of typical underlying functions for a BFN (substitution-permutation functions with any finite number of layers). To do that, for each of them, we first prove tight bounds on the security parameter (number of active S-boxes). Then the security parameter is related to the computational workload of a cipher implementation (modelled as the number of S-boxes computed in the cipher) to obtain an efficiency parameter. Finally, the optimal constructions are those with the maximum resulting efficiency parameter.

The class of ciphers. We focus on balanced Feistel networks with SP-type bijective F-functions, that is, with underlying functions whose internal structure is a substitution-permutation network (SPN). An SPN consists of several sequential applications of an S-box layer (S) – several small nonlinear maps applied in parallel – and a diffusion layer (P) – multiplication by a matrix over a binary finite field. We treat F-functions with $(SP)^{2t}$, $(SP)^{2t+1}$, $(SP)^{2t-1}$, and $(SP)^{2t}$-type F-functions for integer $t \geq 1$. For instance, an $(SP)^{2}$-type F-function consists of two consecutive SP-functions followed by an S-box layer, namely an SPSPS F-function. The instantiation of a Feistel network with an SP-type F-function is deployed in many cryptographic algorithms including E2 [17], TWOFISH [23], CAMELLIA [1], CLEFIA [30], SHA-vide-3 [4], and PICCOLO [25].

Security parameter. Counting the minimum number of active S-boxes is a widely accepted formalism [12] to demonstrate the immunity of a cryptographic algorithm against differential [5] and linear [20] cryptanalysis which are the two most fundamental attacks on block ciphers. Lower bounds on the number of active S-boxes are closely related to the probability of differential trails (under the assumption of subkey independence) and linear trails [12].

For each of the BFN instantiations above, we prove lower bounds on the number of differentially and linearly active S-boxes. In contrast to the previous works [16] and [6], our results with respect to this security parameter:

- generalize the type of the F-function, while [16] and [6] only contain lower bounds for BFNs with SP- and SPS- functions,
- hold for any number of rounds (those of [16] and [6] hold only for a few rounds), and
- contain proofs of tightness for the bounds when the matrices used in the diffusion layers of BFNs are MDS.

Efficiency metric. To measure the efficiency of a construction, we are using the ratio between active S-boxes and all S-boxes in a cipher – a reasonable efficiency metric introduced in [27] and extensively used in [6–9]. It is based on the assumption that most workload one has to perform in the implementation of an SP-type construction is the computation of the S-boxes. Since we are mainly interested in MDS matrices that are equal in all rounds and intend to compare block ciphers of the same block length only, we will ignore the cost of the linear operations for the purposes of comparison.

Optimality. In the wide class of our target ciphers, we prove optimality of several instances with respect to the efficiency parameter. More specifically, among BFN block ciphers with bijective SP-type F-functions and MDS diffusion, we prove BFNs with SPS and SPSP functions to maximize the efficiency in terms of the proportion of active S-boxes in all S-boxes. Interestingly, one SP-layer in the function is not enough to attain optimality, whereas taking more than two S-box layers does not increase the efficiency either.
Organization of the article. The remainder of this article is organized as follows. Section 2 describes the target structure and definitions. The duality of differential and linear trails is explained in Section 3. Section 4 gives proofs for lower bounds on the numbers of differentially and linearly active S-boxes for the BFNs and its results are summarized in Table 1. Section 5 shows the tightness of those bounds. Section 6 discusses the optimality of the BFNs. Finally, we conclude in Section 7.

2 Preliminaries

2.1 Target structures

In this paper, we focus on balanced Feistel networks (BFNs) with bijective F-functions. A $2mn$-bit plaintext $P$ is divided into two subblocks as $P = (X^{(1)}_L, X^{(1)}_U)$, where $X^{(i)}_L, X^{(i)}_U \in \{0, 1\}^{mn}$. Then the $i$-th round output is calculated as follows:

$$(X^{(i+1)}_L, X^{(i+1)}_U) \leftarrow (F(X^{(i)}_L) \oplus X^{(i)}_U, X^{(i)}_L),$$

where $F : \{0, 1\}^{mn} \rightarrow \{0, 1\}^{mn}$ is an F-function in the $i$-th round. A $2mn$-bit ciphertext $C$ for the $r$-round encryption function is derived as $C = (X^{(r+1)}_U, X^{(r+1)}_L)$. Each F-function consists of some S-box layers and linear diffusion layers (P-layers). While $mn$-bit subkeys are XORed before each S-box layer, we omit these subkey additions in this paper for simplicity. An S-box layer consists of $m$ $n$-bit bijective S-boxes, and a linear diffusion layer consists of $mn$-bit linear Boolean function. BFN-(SP)$^u$ denotes BFN with F-functions consisting of $u$ consecutive SP-functions. BFN-(SP)$^u$S denotes BFN with F-functions consisting of $u$ consecutive SP-functions followed by one additional S-box layer. See Figures 1 and 2.

2.2 Notations

We give the standard definitions of bundle weight and branch number followed by more specific notations [12].

**Definition 1 (Bundle Weight)** Let $x \in \{0, 1\}^{mn}$ be represented as $x = (x_1, x_2, ..., x_m)$, where $x_i \in \{0, 1\}^{n}$, then the $n$-bit bundle weight $w_n(x)$ is defined as

$$w_n(x) = \sharp \{i | 1 \leq i \leq m, x_i \neq 0\}. \quad (1)$$
Fig. 2 The \( i \)-th round F-function of BFN-(SP)\(^n\) and BFN-(SP)\(^w\)S.

**Definition 2 (Branch Number)** Let \( M : \{0,1\}^{mn} \rightarrow \{0,1\}^{mn} \). The branch number of \( M \) is defined as

\[
B(M) = \min_{a \neq 0} \{ w_n(a) + w_n(M(a)) \}. \tag{2}
\]

Besides the bundle weight \( w_n \) and branch number \( B \), throughout this paper, we use the following notations:

- \( x_j^{(i)}, y_j^{(i)} \): input and output of the \( j \)-th S-box layer in the \( i \)-th round.
- \( \Delta x_j^{(i)} \): output of the \( j \)-th linear diffusion layer in the \( i \)-th round.
- \( d_j^{(i)} \): a differential trail for \( x_j^{(i)} \).
- \( d_j^{(i)} \): a truncated difference weight of \( x_j^{(i)} \), i.e., \( d_j^{(i)} = w_n(\Delta x_j^{(i)}) \).
- \( d_j^{(i)} \): the number of differentially active S-boxes in the \( i \)-th round.
- \( D(r) \): the minimum number of active S-boxes in \( r \) consecutive rounds.
- \( \Gamma y_j^{(i)} \): a linear trail for \( y_j^{(i)} \).

### 2.3 Efficiency metric

The proportion of active S-boxes in all S-boxes is a reasonable efficiency metric with respect to differential and linear cryptanalysis for ciphers based on substitution-permutation. It was introduced in [27] by Shirai and Preneel for BFNs and used in the works [6–9] for estimating and comparing the efficiency of diverse Feistel constructions, including BFNs.

Both the number of active S-boxes and the number of all S-boxes over several rounds of a BFN depend on the number \( r \) of rounds considered and the number \( m \) of S-boxes in one F-function.

**Definition 3 (\( E_m \))** The efficiency metric \( E_m \) is defined as

\[
E_m = \lim_{r \to \infty} A_{m,r}/S_{m,r},
\]

where \( A_{m,r} \) is the minimum number of active S-boxes over \( r \) rounds and \( S_{m,r} \) is the total number of S-box computations over \( r \) rounds.
Fig. 3 Equivalent transform (BFN-(SP)\textsuperscript{u} to BFN-(PS)\textsuperscript{u}), where thin boxes and thick boxes denote S-box layers and P-layers, respectively.

Note that this efficiency metric \(E_m\) cannot capture all implementation possibilities and constraints in the field, though it is believed to provide an indication of the efficiency of a block cipher towards the two fundamental types of cryptanalysis, see [6–9, 27] for some extensions and discussions with respect to efficiency metrics.

3 Duality of trails

In this section, we demonstrate an equivalence between differential and linear trails for the BFNs. This equivalence follows from Biham’s considerations in [3] and is provided here for completeness. It allows us to work with the minimum numbers of differentially and linearly active S-boxes simultaneously. We first show an equivalent transform for BFN-(SP)\textsuperscript{u}.

**Property 1** Suppose that both S-box layer and linear diffusion layer are bijective. Any BFN consisting of \(u\) consecutive SP-functions, BFN-(SP)\textsuperscript{u}, can be equivalently transformed into a BFN consisting of \(u\) consecutive PS-functions with an initial and a final linear function. This property is seen as the generalization of [16]. Now we have BFN-(PS)\textsuperscript{u} from BFN-(SP)\textsuperscript{u} by using equivalent transforms. An illustration of these equivalent transforms is given in Figure 3. Note that BFN-(PS)\textsuperscript{u} takes \(P' = (P^{-1}(X_L^{(1)}), P^{-1}(X_U^{(1)}))\) as a plaintext and outputs a ciphertext \(C' = (P(X_U^{(r+1)}), P(X_U^{(r+1)})\)). Since these initial and final linear functions do not affect the minimum numbers of active S-boxes, we can ignore these functions when studying the minimum numbers of active S-boxes.

From the concatenation rules [3, 21], \(\Gamma y_1^{(i)} = \Gamma y_u^{(i-1)} \oplus \Gamma y_u^{(i+1)} = \Gamma P(\Gamma x_1^{(i)})\), where \(\Gamma P\) is the bit-based transpose matrix of \(P\). Thus, for BFN-(SP)\textsuperscript{u}, the linear trails can be transformed to the corresponding differential trails by replacing \((\Delta x_1^{(i)}, \Delta x_2^{(i)}, ..., \Delta x_u^{(i)}), (\Delta z_1^{(i)}, \Delta z_2^{(i)}, ..., \Delta z_u^{(i)})\) and \(P\) with \((\Gamma y_1^{(i)}, \Gamma y_u^{(i)}), P\).
..., \Gamma y^{(i)}_u, \Gamma v^{(i)}_u, \ldots, \Gamma v^{(i)}_{u-1}, \ldots, \Gamma v^{(i)}_1)$ and $iP$, respectively. Similarly, for BFN-\((SP)^uS\), the linear trails can be treated as the differential trails by replacing $(\Delta v^{(i)}_u, \Delta y^{(i)}_u, \ldots, \Delta y^{(i)}_{u+1})$ and $P$ by $(\Gamma y^{(i)}_{u+1}, \Gamma y^{(i)}_u, \ldots, \Gamma y^{(i)}_1, \Gamma x^{(i)}_{u+1}, \Gamma x^{(i)}_u, \ldots, \Gamma x^{(i)}_1)$ and $iP$, respectively. Therefore, since the constraints for differential and linear trails for the BFNs are the same, the minimum numbers of differentially and linearly active S-boxes can be derived simultaneously. In the sequel, we only discuss the minimum numbers of differentially active S-boxes for simplicity, keeping in mind, however, that the minimum numbers of linearly active S-boxes can be derived in a similar way.

4 Bounds for active functions

In this section, we give proofs for lower bounds on the minimum number of differentially active S-boxes for BFN-\((SP)^{2t+1}, (SP)^{2t}, (SP)^{2t-1}S\), and \((SP)^tS\). These results are summarized in Table 1.

To prove those bounds, we utilize the following property and lemmata for BFNs consisting of bijective F-functions. The detailed proofs are given in [24].

Property 2 For each nonzero input difference, any two and three consecutive rounds of BFN consisting of bijective F-functions have at least one and two active functions, respectively.

Meanwhile, the numbers of differentially active S-boxes for each differentially active F-function, which is an F-function whose input difference is nonzero, are lower-bounded by the following lemmata.

Lemma 1 (active S-boxes for 1-round BFN-\((SP)^u\)) For BFN-\((SP)^u\), if $d^{(i)}$ is not zero, $d^{(i)} \geq \lceil u/2 \rceil B + (u \mod 2)$.

Recall that $B$ denotes the branch number of the linear layer. Similarly to Lemma 1, one derives the following lemma.

Lemma 2 (active S-boxes for 1-round BFN-\((SP)^uS\)) For BFN-\((SP)^uS\), if $d^{(i)}$ is not zero, $d^{(i)} \geq \lceil u/2 \rceil B + ((u + 1) \mod 2)$.

These lemmata show that the number of active S-boxes can be derived from the number of S-box layers when we treat only one active F-function. However, when we consider some consecutive rounds, the number of active S-boxes does not depend only on the number of S-box layers.

Starting from here, we treat four cases of the F-function construction separately: \((SP)^{2t+1}, (SP)^{2t}, (SP)^{2t-1}S\), and \((SP)^tS\), as those exhibit essential differences.

4.1 Differentially active S-boxes in BFN-\((SP)^{2t+1}\)

For BFN-\((SP)^{2t+1}\), which consists of odd number of SP-layers, the proofs for the lower bounds are the most complicated among other BFNs, since the number of differentially active S-boxes cannot be directly obtained from the number of differentially active F-functions. We find tight lower bounds on the minimum number
of differentially active S-boxes by carefully observing two cases separately: \( t = 0 \) and other cases.

For BFN-(SP)\(^{2t+1}\), Lemma 1 directly translates to the following corollary.

**Corollary 1**

For BFN-(SP)\(^{2t+1}\), if \( d^{(i)} \) is not zero, \( d^{(i)} \geq tB + 1 \).

Property 2 and Corollary 1 directly show that any three consecutive rounds of BFN-(SP)\(^{2t+1}\) have at least \( 2(tB + 1) \) active S-boxes. However, when the center of the F-function in the three consecutive rounds is active, there exist more active S-boxes as follows.

**Lemma 3**

For BFN-(SP)\(^{2t+1}\), if \( d^{(i)} \) is not zero, \( d^{(i-1)} + d^{(i)} + d^{(i+1)} \geq (2t + 1)B \).

The lower bounds on the minimum number of active S-boxes in any consecutive rounds of BFN-(SP)\(^{2t+1}\) are directly derived by the lemmata above. First, we prove the bounds on \( D(r) \), \( r \leq 4 \) by Lemma 4, then show the bounds on \( D(r) \), \( r > 4 \) by Lemma 5.

**Lemma 4**

For BFN-(SP)\(^{2t+1}\), \( D(1) = 0, D(2) = tB + 1, D(3) = 2(tB + 1) \), and \( D(4) = (2t + 1)B \).

The bounds on \( D(r) \), \( r > 4 \), are given as inductive forms.

**Lemma 5**

Let \( r > 4 \), \( D(r) = \min(D(r - 3) + (2t + 1)B, D(r - 4) + (3t + 1)B + 1) \) for BFN-(SP)\(^{2t+1}\).

Now we have the lower bounds in any consecutive rounds of BFN-(SP)\(^{2t+1}\). However, it is hard to compare its efficiency with other constructions, since the bounds are proven as inductive forms. In order to obtain more accurate bounds, we consider two cases. We start with the special case of \( t = 0 \).

**Theorem 1 (active S-boxes for BFN-(SP)\(^{2t+1}, t = 0\)**

For any nonzero input difference (nonzero input mask), every \( 4R, 4R + 1, 4R + 2, 4R + 3 \) rounds of BFN \((R \geq 1)\) with an SP F-function provide at least \((B + 1)R - 1, (B + 1)R, (B + 1)R, (B + 1)R + 1, (B + 1)R + 2\) differentially (linearly) active S-boxes, respectively, assuming \( B > 2 \), where \( B \) is the branch number of the diffusion matrix (of the transposed diffusion matrix).
Note that Theorem 1 was conjectured in [26]. For all other integers \( t > 0 \), the bounds are stated as follows.

**Theorem 2 (active S-boxes for BFN-(SP)\(^{2t+1}\), \( t > 0 \))** For any nonzero input difference (nonzero input mask), every \( 3R \), \( 3R + 1 \), \( 3R + 2 \) rounds of BFN (\( R \geq 1 \)) with \((2t + 1)\) consecutive SP-layers in the F-function (\( t > 0 \)) provide at least \((2t + 1)BR - B + 2\), \((2t + 1)BR\), \((2t + 1)BR + tB + 1\) differentially (linearly) active S-boxes, respectively, where \( B \) is the branch number of the diffusion matrix (of the transposed diffusion matrix).

Now we have comparable bounds for every four rounds of BFN-(SP)\(^{2t+1}\). For the case of \( t > 0 \), \((D(r - 3) + (2t + 1)B)\) is always less than or equal to \((D(r - 4) + (3t + 1)B + 1)\). On the other hands, for the case of \( t = 0 \), \((D(r - 4) + B + 1)\) is less than or equal to \((D(r - 3) + B)\) when \( B > 2 \) and \( r = 4s + 3(s > 0) \) (e.g., \( r = 7, 11, 15, ... \)). Thus, the bounds for the case \( t = 0 \) and \( t > 0 \) are slightly different and those are separately proven. The tightness of these bounds is proven in Section 5.

### 4.2 Differentially active S-boxes in BFN-(SP)\(^{2t}\)

For BFN-(SP)\(^{2t}\), which comprises even number of SP-layers, the minimum number of differentially active S-boxes is straightforwardly proven by observing the number of differentially active F-functions.

Lemma 1 yields the following corollary.

**Corollary 2** For BFN-(SP)\(^{2t}\), if \( d^{(i)} \) is not zero, \( d^{(i)} \geq tB \).

This corollary allows us to prove the following theorem.

**Theorem 3 (active S-boxes for BFN-(SP)\(^{2t}\))** For any nonzero input difference (nonzero input mask), every \( 3R \), \( 3R + 1 \), \( 3R + 2 \) rounds of BFN (\( R \geq 1 \)) with \( 2t \) consecutive SP layers in the F-function provide at least \( 2tBR \), \( 2tBR \), \( 2tBR + tB \) differentially (linearly) active S-boxes, respectively, where \( B \) is the branch number of the diffusion matrix (of the transposed diffusion matrix).

Unlike the case of BFN-(SP)\(^{2t+1}\), the lower bounds for BFN-(SP)\(^{2t}\) are easily proven. In the other words, the minimum number of differentially active S-boxes for BFN-(SP)\(^{2t}\) corresponds to the minimum number of differential active F-functions times \( tB \).

### 4.3 Differentially active S-boxes in BFN-(SP)\(^{2t-1}\)S

Since the number of S-box layers is the same in BFN-(SP)\(^{2t-1}\)S, similarly to the bounds for BFN-(SP)\(^{2t}\), one derives the following theorem.

**Theorem 4 (active S-boxes for BFN-(SP)\(^{2t-1}\)S)** For any nonzero input difference (nonzero input mask), every \( 3R \), \( 3R + 1 \), \( 3R + 2 \) rounds of BFN (\( R \geq 1 \)) with \((2t - 1)\) consecutive SP-layers followed by an S-box layer in the F-function provide at least \( 2tBR \), \( 2tBR \), \( 2tBR + tB \) differentially (linearly) active S-boxes, where \( B \) is the branch number of the diffusion matrix (of the transposed diffusion matrix).
The obtained bounds for BFN-(SP)$^{2t}$S seem almost same as the bounds for BFN-(SP)$^{2t}$. However, BFN-(SP)$^{2t}$ has one more P-layer than BFN-(SP)$^{2t}$S has when the parameter $t$ is the same. This implies that the last P-layer of BFN-(SP)$^{2t}$ does not improve the security in terms of the number of differentially active S-boxes.

4.4 Differentially active S-boxes in BFN-(SP)$^{2t}$S

Similarly to BFN-(SP)$^{2t+1}$, BFN-(SP)$^{2t}$S has odd number of S-layers. However, lack of the last P-layer allows us to prove the bounds for BFN-(SP)$^{2t}$S easily.

Property 2 and Lemma 2 yield the following theorem.

**Theorem 5 (active S-boxes for BFN-(SP)$^{2t}$S)** For any nonzero input difference (nonzero input mask), every $3R, 3R + 1, 3R + 2$ rounds of BFN ($R \geq 1$) with $2t$ consecutive SP-layers followed by an S-box layer in the F-function provide at least $2(tB + 1)R, 2(tB + 1)R, (2(tB + 1)R + (tB + 1))$ differentially (linearly) active S-boxes, where $B$ is the branch number of the diffusion matrix (of the transposed diffusion matrix).

The proof for BFN-(SP)$^{2t}$S is similar to the proofs for BFN-(SP)$^{2t}$ and BFN-(SP)$^{2t-1}$S. In other words, for BFN-(SP)$^{2t}$S, the minimum number of active S-boxes can be proven by studying the number of active F-functions. However, the proven bounds are not same as the bounds for BFN-(SP)$^{2t}$ and BFN-(SP)$^{2t-1}$S, since the number of S-layers is different. In the following sections, we discuss tightness of the bounds proven in this section and their optimality.

5 Tightness of bounds

To demonstrate the tightness of the lower bounds, we provide trails that actually attain those proven bounds when the matrices used in the BFNs are MDS. Figure 4 presents the trail for BFN-SP as an example, which is the most complicated trail among others. The other trails that attain the previously proven bounds are
Table 2 $E_m$ for BFNs with SP-type functions and MDS matrices

<table>
<thead>
<tr>
<th>Construction</th>
<th>$A_{r,m}$</th>
<th>$S_{r,m}$</th>
<th>$E_m = \lim_{r \to \infty} \frac{A_{r,m}}{S_{r,m}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFN-(SP)$^{2t}$</td>
<td>$A_{3R,m} = 2(t(m+1)R)$</td>
<td></td>
<td>$\frac{2tmr}{m+1}$</td>
</tr>
<tr>
<td>BFN-(SP)$^{2t-1}$S</td>
<td>$A_{3R+1,m} = 2t(m+1)R$</td>
<td></td>
<td>$\frac{2tmr}{m+1}$</td>
</tr>
<tr>
<td></td>
<td>$A_{3R+2,m} = (2R+1)(m+1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFN-(SP)$^{2t+1}$</td>
<td>$A_{3R,m} = ((2t+1)R-1)(m+1) + 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_{3R+1,m} = (2t+1)(m+1)R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_{3R+2,m} = (2R+1)(m+1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFN-SP</td>
<td>$A_{4R,m} = (m+2)R - 1$</td>
<td></td>
<td>$\frac{m+2}{4m}$</td>
</tr>
<tr>
<td></td>
<td>$A_{4R+1,m} = 2(2t+1)R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_{4R+2,m} = (2R+1)(m+1)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFN-(SP)$^{2t+2}$</td>
<td>$A_{3R,m} = 2(t(m+1) + 1)R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_{3R+1,m} = 2(t(m+1) + 1)R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A_{3R+2,m} = (2R+1)(t(m+1) + 1)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

omitted due to the page limitation, however, they are given in a similar way. Note that a similar observation for BFN-SP with $m = 8$ was given in Appendix A of [28].

In the figure, $\Delta$ and $\nabla$ denote S-box truncated difference 100...00 (only the first S-box active out of $m$) and 111...11 (all $m$ S-boxes active), respectively. Thin boxes and thick boxes denote S-box layers (S-layers) and linear layers (P-layers), respectively. XORs with difference cancellation are marked with dashed circles. Differentially active S-box layers are denoted by grey. The underlined numbers denote the minimum numbers of active S-boxes in the area indicated by a dashed line.

The paths of Figure 4 for BFN-SP consist of iterative paths and an additional path. In the case of $(4R+3)$ rounds, the tightness is easily proven by the right side of Figure 4. In the other cases $(4R, 4R+1$ and $4R+2$ rounds), paths consist of some consecutive 4-round iterative paths on the left and one 4-round additional path in the center of Figure 4. Each path for $4R$ rounds consists of $(R-1)$ consecutive 4-round iterative paths and one 4-round additional path. Also paths for $4R+1$ and $4R+2$ rounds consist of $R$ consecutive 4-round iterative paths and one 4-round additional path. For example, a path for 12 rounds of BFN-SP consists of two consecutive 4-round iterative paths followed by one 4-round additional path. Similarly, a path for 13 rounds consists of three 4-round iterative paths followed by the first one round of the 4-round additional path.

6 Optimality

In this section, it is proven that BFN-SPS and BFN-SPSP are the most efficient with respect to the efficiency metric $E_m$ of Definition 3. Recall that $E_m$ shows the ratio between active S-boxes and all S-boxes when the number of rounds is sufficiently large. Table 2 contains the computation of $E_m$ for different BFNs in question. The optimality result is formulated as follows.

**Theorem 6** When instantiated with MDS matrices for $m \geq 2$, BFN-(SP)$^{2t}$ and BFN-(SP)$^{2t-1}$S provide a higher or equal proportion of active S-boxes than BFN-SP, BFN-(SP)$^{2t+1}$ and BFN-(SP)$^{2t+2}$S for any number $t$ of layers. Thus, BFN-SPS and BFN-SPSP are optimal with respect to $E_m$. 


7 Conclusion

In this work, we considered a wide class of balanced Feistel networks with any number of interleaved S-box layers and linear diffusion layers in their F-function. In this class, we demonstrated that SPS and SPSP F-functions are arguably optimal with respect to the relative number of active S-boxes provided. Our results indicate that one SP-layer in the F-function is not enough to attain optimality, whereas taking more than two S-box layers does not increase the efficiency either. The optimality is shown with respect to the security of a cipher towards differential and linear cryptanalysis.

As nearly any SPN-based block cipher, BFNs with SP-type F-functions exhibit the differential effect—many differential trails contributing to the same differential. Having SPS or SPSP constructions as F-functions—as in the optimal constructions of this paper—simplifies the consideration of upper bounds on the differential probability over several rounds. The work [2] proves that the maximum average differential probability over 3 rounds for a BFN with bijective F-functions is upper-bounded by \( \pi^2 \), where \( \pi \) is the maximum differential probability of the F-function. At the same time, the maximum differential probability of an SPS or SPSP construction with MDS diffusion is known to be upper-bounded by \( p^m \), where \( p \) is the maximum differential probability of the underlying S-box [18]. This provides an upper bound of \( p^{2m} \) on the average differential probability over 3 rounds of BFN-SPS and BFN-SPSP. Similar applies to the linear probability. However, capturing the differential or linear hull effect for an arbitrary number of rounds and incorporating it into the efficiency metric appears to be a challenging task.

Besides BFNs, generalized Feistel networks (GFNs) are often used in the design of block ciphers. Both CLEFIA [30] and PICCOLO [25] follow this design approach with SP-type F-functions. We conjecture that our optimality result also applies to any GFN under the definition of [10]. In other words, our conjecture is that the instantiation of the F-function with SPS and SPSP will be optimal with respect to the relative number of active S-boxes. We leave this as an important open problem.

References

24. K. Shibutani, A. Bogdanov: Towards the Optimality of Feistel Ciphers with Substitution-Permutation Functions (full paper).
Type 1.x Generalized Feistel Structures

Shingo Yanagihara · Tetsu Iwata

Received: date / Accepted: date

Abstract We formalize the Type 1.x Generalized Feistel Structure (GFS) in order to fill the gap between Type 1 and Type 2 GFSs. This is a natural extension of Type 1 and Type 2 GFSs, and covers them as special cases. The diffusion property of GFS is known to vary depending on the permutation used in the round function. When we have two non-linear functions in one round, we propose a permutation that has a good diffusion property, and compare it with the structure that uses a sub-block-wise cyclic shift. We also present experimental results of exhaustively evaluating the diffusion properties of all permutations up to eight sub-blocks.

Keywords Blockcipher · Type 1.x generalized Feistel structure · permutation layer.

1 Introduction

Background. The Generalized Feistel Structure, which we write GFS, is one of the structures used in the design of blockciphers and hash functions. In the classical Feistel structure, the plaintext is divided into two sub-blocks, and these are encrypted through several round functions, while in GFS, the plaintext is divided into $d$ ($\geq 3$) sub-blocks. Compared to the SP network used in AES [4], GFS has an advantage in that computation of the nonlinear functions is the same in encryption and decryption. On the other hand, the diffusion property of GFS is generally poor and it requires many rounds to be secure.

There are several types of GFSs [12, 5]. For example, Type 1, Type 2, Type 3, Source-Heavy, Target-Heavy, Alternating, and Nyberg’s GFSs are known. We will focus on Type 1 and Type 2 GFSs, and the former is used for example in CAST-256 and Lesamnta, and the latter is used in RC6, HIGHT, and CLEFIA.

S. Yanagihara
Nagoya University, Japan
E-mail: s.yanagi@echo.nuee.nagoya-u.ac.jp

T. Iwata
Nagoya University, Japan
E-mail: iwata@cse.nagoya-u.ac.jp
The security of these structures has been extensively evaluated. [5] shows the security of Nyberg’s GFS. The lower bounds on the number of active S-boxes for Type 1 and Type 2 GFSs with single SP-functions and single-round diffusion are shown in [10] and [6], respectively. [7] derives the lower bounds in the case of using multiple-round diffusions, and [1,3,2] analyze the security of Type 1 and Type 2 GFSs with double SP-functions.

In the round function, GFS generally uses the sub-block-wise cyclic shift over the sub-blocks, which we write \( \pi_s \), as the permutation. As stated above, the diffusion of GFS is generally slow and it requires many rounds to be secure. At FSE 2010, Suzaki and Minematsu proposed to use a non-cyclic shift in Type 2 GFS, and demonstrated that the diffusion property is improved by changing the permutation from \( \pi_s \) [8]. The result is incorporated into the design of a practical blockcipher called TWINE [9]. Following [8], Yanagihara and Iwata studied the diffusion properties of Type 1, Type 3, Source-Heavy, and Target-Heavy GFSs [11]. They showed that the diffusion properties of Type 1 and Type 3 GFSs can be improved by changing the permutation from \( \pi_s \).

In this paper, we abuse the definition of GFS and we use the term GFS to mean structures where their permutations are not restricted to the sub-block-wise cyclic shift. Now Type 1 GFS has one non-linear function in one round function, while Type 2 GFS has \( d/2 \) functions, where \( d \) is the number of sub-blocks which is even in Type 2 GFS. Then a natural question is to see structures where we have \( \eta \) non-linear functions in the round function, where \( 1 \leq \eta \leq d/2 \).

Our Contributions. In this paper, we formalize a type of GFS called Type 1.x GFS to fill the gap between Type 1 and Type 2 GFSs. Type 1.x GFS is characterized by two integers \( d \) and \( \eta \), where \( d \) is the number of sub-blocks which is not necessarily even, and \( \eta \) is the number of the non-linear functions in one round function. Type 1.x GFS covers Type 1 and 2 GFSs as special cases; they correspond to \( \eta = 1 \) and \( \eta = d/2 \), respectively. We note that Type 1.x GFS itself is not new. For example the key schedule of 80-bit key TWINE [9] can be seen as this structure with \( d = 20 \) and \( \eta = 2 \).

For \( \eta = 2 \), we propose a construction of a permutation, which we write \( \pi_p \), that has a good diffusion property. We also characterize the diffusion property of Type 1.x GFS when we use \( \pi_s \) as the permutation. We compare these two values to see that \( \pi_p \) indeed has a better diffusion property. We also present experimental results of exhaustively evaluating the diffusion properties of all permutations for parameters \( 3 \leq d \leq 8 \) and \( 1 \leq \eta \leq \lfloor d/2 \rfloor \), and confirm that the proposed construction is the optimum construction in terms of diffusion.

2 Preliminaries

We first introduce the Generalized Feistel Structure (GFS). Let \( n \) and \( d \) be integers, where \( n \) is the size of a sub-block in bits and \( d \) is the number of sub-blocks. Let \( x^0 = (x_0^0, x_1^0, \ldots, x_{d-1}^0) \in (\{0,1\}^n)^d \) be a \( dn \)-bit plaintext. GFS encrypts \( x^0 \) with a secret key by applying the round function for \( R \) times iteratively, and outputs a \( dn \)-bit ciphertext \( x^R = (x_0^R, x_1^R, \ldots, x_{d-1}^R) \in (\{0,1\}^n)^d \).

In Fig. 1, we show an example of the round function. The round function consists of \( F \)-Layer and \( II \)-Layer. \( F \)-Layer has key dependent \( F \) functions and the
xor operations. The structure of $F$-Layer depends on the types of GFS. II-Layer is a permutation $\pi$ over $d$ sub-blocks. We assume that in the final round, the round function consists of only $F$-Layer so that both encryption and decryption start with $F$-Layer. Similarly to encryption, decryption is done by using $F^{-1}$-Layer and II$^{-1}$-Layer instead of $F$-Layer and II-Layer.

As shown in Fig. 1, the $d$ sub-blocks are sequentially numbered from left to right as $0, 1, \ldots, d-1$ and $\pi$ is considered to be a permutation over $\{0, 1, \ldots, d-1\}$. We write $\pi(i)$ for the index of the sub-block after applying $\pi$ to the $i$-th sub-block. For example, in Fig. 1, we have $\pi(0) = 3, \pi(1) = 0, \pi(2) = 1$, and $\pi(3) = 0$, and we also write them as $\pi = (3, 0, 1, 2)$ collectively. For an integer $r \geq 1$, $\pi^r(i)$ is the index of the sub-block after applying $\pi$ to the $i$-th sub-block for $r$ times. Similarly, $\pi^{-r}(i)$ is the index of the sub-block after applying $\pi^{-1}$ to the $i$-th sub-block for $r$ times. We treat $\pi^0(i)$ as $i$. We define $\pi_s$ as $\pi_s = (d-1, 0, 1, \ldots, d-2)$, i.e. $\pi_s$ is the sub-block-wise left cyclic shift.

For encryption, let $x^r$ be the intermediate result after encrypting $r$ rounds, and $x^r_i$ be the $i$-th sub-block of $x^r$. We write $x^r = (x^r_0, \ldots, x^r_{d-1}) \in (\{0, 1\}^n)^d$. For decryption, we define $y^r$ and $y^r_i$ analogously. From these definitions, $x^0$ and $y^R$ correspond to the plaintext, and $y^0$ and $x^R$ correspond to the ciphertext.

Next, we introduce Type 1 and Type 2 GFSs.

**Type 1 GFS.** We write $E_{T1}(\pi)$ for Type 1 GFS that uses $\pi$ in II-Layer. For $E_{T1}(\pi)$, $x^r_i$ is defined as

$$x^r_i = \begin{cases} 
F(x^r_{i-1}) \oplus x^r_{i-1} & \text{if } \pi^{-1}(i) = 1, \\
-x^r_{\pi^{-1}(i)} & \text{otherwise},
\end{cases}$$

where $F : \{0, 1\}^n \to \{0, 1\}^n$. See Fig. 2 for an example of $E_{T1}(\pi_s)$ with $d = 6$. 
where \( F : \{0, 1\}^n \to \{0, 1\}^n \). Type 2 GFS is not defined when \( d \) is odd. See Fig. 3 for an example of \( E^{T2}(\pi_s) \) with \( d = 6 \).

Type 3 [12], Source-Heavy, Target-Heavy, and Alternating are also known as other types of GFS. We note that as we allow any permutation in \( \Pi \)-Layer, Nyberg’s GFS can be seen as a special case of Type 2 GFS.

3 Type 1.x GFS

Definition of Type 1.x GFS. In this section, we formalize a type of GFS which we call Type 1.x GFS.

Definition 1 Let \( d \) and \( \eta \) be integers such that \( d \geq 3 \) and \( 1 \leq \eta \leq [d/2] \). In Type 1.x GFS, \( x_i^n \) is defined as

\[
x_i^n = \begin{cases} 
F(x_{\pi^{-1}(i)-1}^{r-1}) \oplus x_{\pi^{-1}(i)}^{r-1} & \text{if } \pi^{-1}(i) \mod 2 = 1 \text{ and } \pi^{-1}(i) \leq 2\eta - 1, \\
x_{\pi^{-1}(i)}^{r-1} & \text{otherwise},
\end{cases}
\]

where \( F : \{0, 1\}^n \to \{0, 1\}^n \). We write \( E^{(d, \eta)}(\pi) \) for Type 1.x GFS that uses \( \pi \) in \( \Pi \)-Layer.

Figure 4 shows \( E^{(6,2)}(\pi_s) \). We see that \( E^{T1}(\pi) \) with \( d \) sub-blocks is equivalent to \( E^{(d,1)}(\pi) \), and \( E^{T2}(\pi) \) with \( d \) sub-blocks is equivalent to \( E^{(d,d/2)}(\pi) \) where \( d \) is even. For example, Fig. 2 is \( E^{(6,1)}(\pi_s) \) and Fig. 3 is \( E^{(6,3)}(\pi_s) \). The integer, \( \eta \), denotes the number of \( F \) functions in one round, which is equivalent to the number of \( \oplus \) operations in one round.

Equivalence of Type 1.x GFS. We next define the equivalence of Type 1.x GFS. Intuitively, if \( E^{(d, \eta)}(\pi') \) is obtained by exchanging several sub-blocks in \( E^{(d, \eta)}(\pi) \), then we say that these two structures are equivalent.

More precisely, we say that \( E^{(d, \eta)}(\pi) \) and \( E^{(d, \eta)}(\pi') \) are equivalent if there exists \( \pi' = (a_0, \ldots, a_{d-1}) \) such that \( \pi' = \pi \circ \pi \circ (\pi')^{-1} \), where

\[
\begin{align*}
\{a_0, a_2, \ldots, a_{2\eta-2}\} &= \{0, 2, \ldots, 2\eta - 2\}, \\
a_1 &= a_0 + 1, a_3 = a_2 + 1, \ldots, a_{2\eta-1} = a_{2\eta-2} + 1, \text{and} \\
\{a_{2\eta}, a_{2\eta+1}, \ldots, a_{d-1}\} &= \{2\eta, 2\eta + 1, \ldots, d - 1\}.
\end{align*}
\]

We note that \( g \circ f(x) \) is \( g(f(x)) \). That is, \( E^{(d, \eta)}(\pi) \) and \( E^{(d, \eta)}(\pi') \) are equivalent if \( \pi' \) permutes the elements within \( \{(0, 1), (2, 3), \ldots, (2\eta-2, 2\eta-1)\} \) and \( \{2\eta, 2\eta + 1, \ldots, d - 1\} \). It can be verified that \( E^{(d, \eta)}(\pi') = \pi' \circ E^{(d, \eta)}(\pi) \circ (\pi')^{-1} \). Therefore, \( E^{(d, \eta)}(\pi) \) and \( E^{(d, \eta)}(\pi') \) are the same structures except that the orders of the input and the output are different.
4 Diffusion Property

In this section, following [8], we introduce the notion of \( \text{DR}_{\text{max}}^{(d, \eta)}(\pi) \) to evaluate the diffusion property of Type 1.x GFS.

**Data Dependent Variables.** We first introduce the *data dependent variables* of Type 1.x GFS, which we write \( X^r \) and \( Y^r \). For encryption, if we consider the diffusion of the \( i \)-th sub-block of the plaintext, we let \( X^0 = (X^0_0, X^0_1, \ldots, X^0_{d-1}) \in \{0, 1\}^d \), where \( X^0_i = 1 \) and \( X^0_i = 0 \) for \( \forall i \neq i \). Intuitively, if the \( i \)-th input sub-block diffuses to the \( j \)-th sub-block after encrypting \( r \) rounds, then we let \( X^r_j = 1 \), and \( X^r_j = 0 \) otherwise. We write \( X^r = (X^r_0, X^r_1, \ldots, X^r_{d-1}) \). Similarly, the data dependent variable \( Y^r = (Y^r_0, Y^r_1, \ldots, Y^r_{d-1}) \in \{0, 1\}^d \) is defined for decryption. If \( Y^r_0 = 1 \) and \( Y^r_j \) depends on \( Y^r_0 \), then we let \( Y^r_j = 1 \), and \( Y^r_j = 0 \) otherwise.

Let \( j = \pi^{-1}(i) \). If \( j \mod 2 = 1 \) and \( j \leq 2\eta - 1 \), then the \( i \)-th sub-block of the \( r \)-th round depends on the \((j - 1)\)-st sub-block through the \( F \) function and the \( j \)-th sub-block of the \((r - 1)\)-st round. Otherwise, the \( i \)-th sub-block of the \( r \)-th round depends on the \( j \)-th sub-block of the \((r - 1)\)-st round. Therefore, given \( X^0 = (X^0_0, X^0_1, \ldots, X^0_{d-1}) \in \{0, 1\}^d \), \( X^r \) for \( r \geq 1 \) is successively defined as follows.

\[
X^r_i = \begin{cases} 
X^r_{\pi^{-1}(i)-1} \lor X^r_{\pi^{-1}(i)} & \text{if } \pi^{-1}(i) \mod 2 = 1 \text{ and } \pi^{-1}(i) \leq 2\eta - 1, \\
X^r_{\pi^{-1}(i)} & \text{otherwise.}
\end{cases}
\]

We note that \( a \lor b \) is the or operation of \( a \) and \( b \). Similarly, for decryption, given \( Y^0 = (Y^0_0, Y^0_1, \ldots, Y^0_{d-1}) \in \{0, 1\}^d \), \( Y^r \) for \( r \geq 1 \) is defined as follows.

\[
Y^r_i = \begin{cases} 
Y^r_{\pi(i)-1} \lor Y^r_{\pi(i)} & \text{if } \pi(i) \mod 2 = 1 \text{ and } \pi(i) \leq 2\eta - 1, \\
Y^r_{\pi(i)} & \text{otherwise.}
\end{cases}
\]

Let \( |X^r| \) be the number of bit “1” in \( X^r \). If \( X^r_i = 1 \), then we say that the \( i \)-th sub-block is *active*. If there exists \( r \geq 0 \) such that \( |X^r| = d \), then we say that the input \( X^0 \) achieves FD (Full Diffusion). If all \( X^0 \) such that \( |X^0| = 1 \) achieve FD, then we say that the GFS achieves FD.

**Definition of \( \text{DR}_{\text{max}}^{(d, \eta)}(\pi) \) [8].** Next, we define \( \text{DR}_{\text{max}}^{(d, \eta)}(\pi) \), which is used to characterize the diffusion property of Type 1.x GFS using \( \pi \) in \( H \)-layer. This value is defined as the minimum number of round so that every sub-block depends on all the input sub-blocks.

More precisely, it is defined as follows:

\[
\text{DR}_{\text{max}}^{(d, \eta)}(\pi) \overset{\text{def}}{=} \max\{\text{DR}_{\text{max}}^{(d, \eta)}_E(\pi), \text{DR}_{\text{max}}^{(d, \eta)}_D(\pi)\}
\]

First, we define \( \text{DR}_{\text{max}}^{(d, \eta)}_E(\pi) \) by using \( \text{DR}_{\text{max}}^{(d, \eta)}_{E,i}(\pi) \). \( \text{DR}_{\text{max}}^{(d, \eta)}_{E,i}(\pi) \) is the minimum number of round such that the \( i \)-th input sub-block diffuses to all the sub-blocks in the encryption direction. Therefore, \( \text{DR}_{\text{max}}^{(d, \eta)}_{E,i}(\pi) \) is defined as

\[
\text{DR}_{\text{max}}^{(d, \eta)}_{E,i}(\pi) \overset{\text{def}}{=} \min\{r \mid \forall i', \exists i, X^0_i = 0, X^0_i = 1, |X^r| = d\}.
\]
Then \( \text{DRmax}_{E}^{(d, \eta)}(\pi) \) is defined as the maximum of \( \text{DRmax}_{E,i}^{(d, \eta)}(\pi) \) over all \( 0 \leq i \leq d - 1 \) as follows:

\[
\text{DRmax}_{E}^{(d, \eta)}(\pi) \overset{\text{def}}{=} \max\{\text{DRmax}_{E,i}^{(d, \eta)}(\pi) \mid 0 \leq i \leq d - 1\}
\]

Similarly, \( \text{DRmax}_{D,i}^{(d, \eta)}(\pi) \) and \( \text{DRmax}_{D}^{(d, \eta)}(\pi) \) are defined for decryption. We let \( \text{DRmax}_{D,i}^{(d, \eta)}(\pi) \overset{\text{def}}{=} \min\{|r| \mid \forall i' \neq i, Y_{i'}^{0} = 0, Y_{i}^{0} = 1, |Y_{r}| = d \} \) and \( \text{DRmax}_{D}^{(d, \eta)}(\pi) \overset{\text{def}}{=} \max\{\text{DRmax}_{D,i}^{(d, \eta)}(\pi) \mid 0 \leq i \leq d - 1\} \).

We note that \( E^{(d, \eta)}(\pi) \) has a better diffusion property if \( \text{DRmax}_{E,i}^{(d, \eta)}(\pi) \) is smaller, and \( \text{DRmax}_{D}^{(d, \eta)}(\pi) = \infty \) implies that there exists an input sub-block \( x_{i}^{0} \) such that some output sub-block is independent of \( x_{i}^{0} \) after any number of rounds. We also remark that the above definitions are given for the sub-block-wise dependency, and that “linear dependency” is sufficient. That is, the definition of FD does not mean that every output bit depends on all the input bits.

There are \( 6! \) permutations over \( \{0, 1, \ldots, d - 1\} \) in total, and we say that the permutation \( \pi \) is optimum in terms of diffusion if \( \text{DRmax}_{E}^{(d, \eta)}(\pi) \) is the minimum among all the \( 6! \) permutations. We note that the optimum \( \pi \) may not be unique.

We also note that this paper focuses on the diffusion property of a construction and we use \( \text{DRmax}_{E}^{(d, \eta)}(\pi) \) to characterize the diffusion property. We expect that a better diffusion property implies better resistance against various cryptographic attacks, e.g., [8] shows, under a certain condition on the permutation, the relation between \( \text{DRmax}_{E}^{(d,d/2)}(\pi) \) and the number of rounds of the impossible differential path and saturation path. However, \( \text{DRmax}_{E}^{(d, \eta)}(\pi) \) may or may not directly reflect the security against, for example, differential or linear attacks, and hence the security against these attacks has to be evaluated independently.

\section{Proposed Construction \( \pi_{p} \)}

In this section, we focus on \( \eta = 2 \) and present our proposed construction which we call \( \pi_{p} \). We also derive a value of \( \text{DRmax}^{(d,2)}(\pi_{p}) \).

First, we present our construction of \( \pi_{p} \).

\textbf{Definition 2} Let \( d \geq 5 \), and \( a \) be an integer such that \( 1 \leq a \leq d - 3 \). We define \( \pi_{p} \) as follows:

\[
\pi_{p} = \begin{cases} 
(1, 3, 4, 2, 5, 6, \ldots, d - 1, 0) & \text{if } a = 1 \\
(1, 4, 0, 2, 5, 6, \ldots, d - 1, 3) & \text{if } a = d - 3 \\
(1, 4, a + 3, 2, 5, 6, \ldots, a + 2, 3, a + 4, a + 5, \ldots, d - 1, 0) & \text{otherwise}
\end{cases}
\]

For example, we have the following \( \pi_{p} \)’s when \( d = 9 \):

\[
\pi_{p} = \begin{cases} 
(1, 3, 4, 2, 5, 6, 7, 8, 0) & \text{if } a = 1 \\
(1, 4, 5, 2, 3, 6, 7, 8, 0) & \text{if } a = 2 \\
(1, 4, 6, 2, 5, 3, 7, 8, 0) & \text{if } a = 3 \\
(1, 4, 7, 2, 5, 6, 3, 8, 0) & \text{if } a = 4 \\
(1, 4, 8, 2, 5, 6, 7, 3, 0) & \text{if } a = 5 \\
(1, 4, 0, 2, 5, 6, 7, 8, 3) & \text{if } a = 6
\end{cases}
\]
Type 1.x Generalized Feistel Structures

See Fig. 6 for an example of Type 1.x GFS with $\pi_p$ when $(d, \eta) = (9, 2)$ and $a = 1$.

We next introduce $r_{ij}$ and $SB_{ij}$ which are used to compute $\text{DRmax}^{(d,2)}(\pi_p)$.

**Definition 3** For any $\pi$, let $r_{ij}$ be the smallest integer $r \geq 1$ such that $\pi^r(i) = j$, and a set $SB_{ij}$ be $SB_{ij} = \{k | \pi^r(i) = k, 0 < r \leq r_{ij}\}$.

Intuitively, $r_{ij}$ is the smallest number of applications of $\pi$ so that the $i$-th sub-block propagates to the $j$-th sub-block, and $SB_{ij}$ is the set of sub-blocks which are passed in the process.

Figure 6 shows the propagation of the left four sub-blocks of Type 1.x GFS with $\eta = 2$ and $\pi_p$. The 0th sub-block propagates to the 1st sub-block, and the 3rd sub-block propagates to the 2nd sub-block in one round. The 1st sub-block propagates to the 3rd sub-block in $a$ rounds, and the 3rd sub-block propagates to the 2nd sub-block in $(d - 2 - a)$ rounds. By using the notation of $r_{ij}$ and $SB_{ij}$, we have

$$
\begin{align*}
\begin{cases}
0_1 = r_{32} = 1, r_{13} = a, r_{20} = d - 2 - a, \\
0, 2 \notin SB_{13}, 1 \notin SB_{20}, SB_{13} \cap SB_{20} = \emptyset, \text{and} \\
SB_{13} \cup SB_{20} \cup \{1, 2\} = \{0, 1, \ldots, d - 1\},
\end{cases}
\end{align*}
$$

(1)

since $SB_{13} = \{3, 4, 5, \ldots, a + 1, a + 2\}$ and $SB_{20} = \{0, a + 3, a + 4, \ldots, d - 1\}$.

Intuitively, $\pi_p$ is designed so that if the 2nd sub-block of $E^{(d,2)}(\pi_p)$ is active in the encryption direction, then all the subsequent 2nd sub-blocks remain active as well. Similarly, if the 1st sub-block is active in the decryption direction, then all the 1st sub-blocks in the following rounds are active. The right $(d - 2\eta)$ sub-blocks are used as the propagation from the 1st and 2nd sub-blocks to the 3rd and 0th sub-blocks, respectively.

The next lemma shows the value of $\text{DRmax}^{(d,2)}(\pi_p)$.

**Lemma 1** Let $d \geq 5$. Then we have $\text{DRmax}^{(d,2)}(\pi_p) = 2d - 4$. 

For decryption, we have $\text{DR}_{\pi_{i+1}}^{(d,2)}(\pi_p) = 2(r_{i+3} + r_{20}) = 2d - 4$ is larger than other $\text{DR}_{E,\pi_{i+1}}^{(d,2)}(\pi_p)$’s. For decryption, we have $\text{DR}_{\pi_{i+2}}^{(d,2)}(\pi_p) = \text{DR}_{E,\pi_{i+2}}^{(d,2)}(\pi_p)$ since $E^{(d,2)}(\pi_p)$ and $E^{(d,2)}(\pi_{i+2})$ are equivalent, and the lemma follows.

Lemma 1 covers $d \geq 5$. For $d = 4$, we can define $\pi_p = (1, 3, 0, 2)$. Then we have $r_{01} = r_{32} = r_{13} = r_{20} = 1$. Since $SB_{13} = \{3\}$ and $SB_{20} = \{0\}$, we also have $0, 2 \notin SB_{13}, 1 \notin SB_{20}, SB_{13} \cap SB_{20} = \emptyset$, and $SB_{13} \cup SB_{20} \cup \{1, 2\} = \{0, 1, 2, 3\}$. Since these properties are equivalent to (1) and the proof of Lemma 1 works with these properties, we have $\text{DR}_{\pi_{i+1}}^{(d,2)}(\pi_p) = 2 \times 4 - 4 = 4$.

6 Analysis of $E^{(d, \eta)}(\pi_s)$

In this section, we analyze the diffusion property of $E^{(d, \eta)}(\pi_s)$.

**Lemma 2** For any $d \geq 3$ and $1 \leq \eta \leq \lfloor d/2 \rfloor$, we have $\text{DR}_{E,\pi_{i+1}}^{(d, \eta)}(\pi_s) = \max\{\text{DR}_{E,\pi_{i+2}}^{(d, \eta)}(\pi_s), \text{DR}_{E,\pi_{i+3}}^{(d, \eta)}(\pi_s)\}$, where

\[
\text{DR}_{E,\pi_{i+2}}^{(d, \eta)}(\pi_s) = \begin{cases} 
\left\lfloor \frac{d - \eta}{\eta} \right\rfloor (d - \eta) + 2 & \text{if } (d - 2) \mod \eta = 0 \\
\left\lfloor \frac{d - \eta}{\eta} \right\rfloor (d - \eta) + 2(d - \eta) & \text{otherwise}
\end{cases}
\]

\[
\text{DR}_{E,\pi_{i+3}}^{(d, \eta)}(\pi_s) = \begin{cases} 
\left\lfloor \frac{d - 1}{\eta} \right\rfloor (d - \eta) + 1 & \text{if } (d - 1) \mod \eta = 0 \\
\left\lfloor \frac{d - 1}{\eta} \right\rfloor (d - \eta) + 2(d - \eta) & \text{otherwise}
\end{cases}
\]

We show a brief overview of the proof. First, we compare $\text{DR}_{E,\pi_{i+1}}^{(d, \eta)}(\pi_s)$ for all $0 \leq i \leq d - 1$, and show that $\text{DR}_{E,\pi_{d-1}}^{(d, \eta)}(\pi_s) = 2d - 2\eta$ is the largest. Similarly, we analyze the decryption direction, and we have (2) and (3). Then we show that $\text{DR}_{E,\pi_{2d-3}}^{(d, \eta)}(\pi_s)$ and $\text{DR}_{E,\pi_{2d-1}}^{(d, \eta)}(\pi_s)$ are the candidates for the largest $\text{DR}_{E,\pi_{d-1}}^{(d, \eta)}(\pi_s)$ value. Finally, we compare these three values to show $\text{DR}_{E,\pi_{2d-2}}^{(d, \eta)}(\pi_s) \geq \text{DR}_{E,\pi_{2d-1}}^{(d, \eta)}(\pi_s)$ and $\text{DR}_{E,\pi_{2d-3}}^{(d, \eta)}(\pi_s) \geq \text{DR}_{E,\pi_{d-1}}^{(d, \eta)}(\pi_s)$, and we have the lemma.

**Discussions.** We summarize the comparison between Lemma 1 and Lemma 2 for $3 \leq d \leq 16$ in Fig. 7. From the figure, we see that $E^{(d,2)}(\pi_p)$ is better than $E^{(d,2)}(\pi_s)$ in terms of the diffusion property for $3 \leq d \leq 16$, that is, we have $\text{DR}_{E,\pi_{d}}^{(d,2)}(\pi_s) \geq 2(d - 2) = 2d - 4 = \text{DR}_{E,\pi_{d}}^{(d,2)}(\pi_p)$. We note that we have experimentally verified the result.

7 Experimental Results

In this section, we present our experimental results on computing $\text{DR}_{E,\pi}^{(d, \eta)}(\pi)$. We compute $\text{DR}_{E,\pi}^{(d, \eta)}(\pi)$ for all $3 \leq d \leq 8$ and $1 \leq \eta \leq \lfloor d/2 \rfloor$. In Tables 1–4,
we list $\pi_s$ and all optimum $\pi$'s, i.e., $\text{DRmax}^{(d,n)}(\pi)$ is the minimum among all permutations for given $d$ and $\eta$. Only the lexicographically first permutations in the equivalent classes are presented in the tables. Table 1 shows the results for $\eta = 1$. Similarly, Tables 2, 3, and 4 show results for $\eta = 2, 3,$ and $4$, respectively.

In the tables, permutations with $s$ are equivalent to $\pi_s$. In Table 2 (for $\eta = 2$), the superscript in permutations with $p$ indicates a value of $a$ in Definition 2.

Table 1 corresponds to Type 1 GFS, and we see that the result is the same as the result in [11]. Also, we can verify that when $d = 6$, the permutation $\pi = (1, 2, 5, 0, 3, 4)$ in Table 3 is equivalent to the inverse of “No. 1” in [8, p. 38, $k = 6$]. Similarly, when $d = 8$, $\pi = (3, 0, 1, 4, 7, 2, 5, 6)$ and $\pi = (1, 2, 5, 0, 3, 6, 7, 4)$ in Table 4 are equivalent to “No. 1” and its inverse in [8, p. 38, $k = 8$], respectively. Furthermore, $\pi = (1, 2, 5, 6, 7, 4, 3, 0)$ in Table 4 is equivalent to both “No. 2” and its inverse in [8, p. 38, $k = 8$].

We summarize observations from these tables as follows:

- In many cases, there exist optimum permutations $\pi$ such that $\text{DRmax}^{(d,n)}(\pi) < \text{DRmax}^{(d,n)}(\pi_s)$, i.e., the diffusion property of Type 1.x GFS can be improved by changing the permutation from $\pi_s$.
- For $4 \leq d \leq 8$ and $\eta = 2$, the proposed construction $\pi_p$ is the optimum permutation in terms of diffusion.

For given $d$ and $\eta$, Tables 1–4 are useful to know a permutation with a good diffusion property. However, suppose that $(d, \eta) \neq (d', \eta')$ and we want to compare the diffusion properties of $E^{(d,n)}(\pi)$ and $E^{(d',n')}(\pi')$. In this case, the value of DRmax may not be suitable for this purpose, and one possible option to make a comparison is to consider the value $\eta \text{DRmax}^{(d,n)}(\pi)/d$, which can be easily derived from Tables 1–4.
hence the structures obtained from Tables 1–4 do not imply that they can be used for a good diffusion property. Also, we have focused on the diffusion property and we have shown that Type 1.x GFS with $\text{DR}_{\text{max}}$ is better than the one with $\pi_s$ in terms of diffusion. Finally, we presented our experimental results for $3 \leq d \leq 8$ and $1 \leq \eta \leq \lfloor d/2 \rfloor$.

As future work, it would be interesting to see constructions for $\eta > 2$ with a good diffusion property. Also, we have focused on the diffusion property and hence the structures obtained from Tables 1–4 do not imply that they can be used for a good diffusion property.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$(2, 0, 1)_{r}$</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>$(2, 0, 0, 3, 1)_{r}$</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>$(2, 0, 3, 4, 1)_{r}$</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>$(2, 0, 3, 4, 1, 1)_{r}$</td>
<td>26</td>
</tr>
<tr>
<td>7</td>
<td>$(2, 0, 3, 4, 5, 6, 1)_{r}$</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>$(2, 0, 3, 4, 5, 6, 7, 1)_{r}$</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$(1, 4, 0, 2)_{p}$</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>$(1, 4, 0, 2, 0)_{p}$</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>$(1, 4, 0, 2, 0, 2)_{p}$</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>$(1, 4, 0, 2, 5, 6, 3)_{p}$</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>$(1, 4, 0, 2, 5, 6, 7, 0)_{p}$</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>$(1, 4, 0, 2, 5, 6, 7, 1)_{p}$</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$(1, 2, 0, 4, 5, 8)_{s}$</td>
<td>5 [8]</td>
</tr>
<tr>
<td>7</td>
<td>$(1, 2, 0, 4, 5, 6, 3)_{s}$</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 3)_{s}$</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$(1, 2, 0, 4, 5, 6, 3)_{s}$</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 3)_{s}$</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 1)_{s}$</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 0)_{s}$</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$(1, 2, 0, 4, 5, 6, 3)_{s}$</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 3)_{s}$</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 1)_{s}$</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$\pi$</th>
<th>$\text{DR}_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>$(1, 2, 0, 4, 5, 6, 3)_{s}$</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 3)_{s}$</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>$(1, 2, 0, 4, 5, 6, 7, 1)_{s}$</td>
<td>14</td>
</tr>
</tbody>
</table>

8 Conclusions

In this paper, we formalized Type 1.x GFS, which covers Type 1 and Type 2 GFSs as special cases. We also proposed a construction $\pi_p$ for $\eta = 2$ and derived $\text{DR}_{\text{max}}(d, 2)(\pi_p)$. We then derived $\text{DR}_{\text{max}}(d, 0)(\pi_s)$, and we showed that Type 1.x GFS with $\pi_s$ is better than the one with $\pi_s$ in terms of diffusion. Finally, we presented our experimental results for $3 \leq d \leq 8$ and $1 \leq \eta \leq \lfloor d/2 \rfloor$.

As future work, it would be interesting to see constructions for $\eta > 2$ with a good diffusion property. Also, we have focused on the diffusion property and hence the structures obtained from Tables 1–4 do not imply that they can be used for a good diffusion property.
in practice with the suggested number of rounds. The security against various cryptographic attacks remains to be evaluated.

Acknowledgements The authors would like to thank the reviewers for useful comments. This work was supported in part by MEXT KAKENHI, Grant-in-Aid for Young Scientists (A), 22680001.

References

Paillier-Based Publicly Verifiable (Non-interactive) Secret Sharing

Mahabir Prasad Jhanwar · Ayineedi Venkateswarlu · Reihaneh Safavi-Naini

Abstract A publicly verifiable secret sharing (PVSS) scheme is a verifiable secret sharing (VSS) scheme in which anyone, not only the shareholders, can verify that the secret shares are correctly distributed. PVSS plays an essential role in the systems that use VSS. In this paper we present a new construction for PVSS scheme based on Paillier encryption scheme. We formalize the notion of indistinguishability of secrets and prove that our scheme achieves it under the Decisional Composite Residuosity Assumption (DCRA).

Keywords Secret sharing · non-interactive PVSS · Paillier’s encryption scheme · Decisional Composite Residuosity Assumption

1 Introduction

(Verifiable) Secret Sharing is one of the most important tools in modern cryptography. The concept and the first realization of secret sharing were presented independently in [22] and in [3]. Since then much work has been put into the investigation of such schemes. In a secret sharing scheme, there exists a dealer and \(\ell\) participants. The dealer splits a secret, say \(s\), into \(\ell\) different pieces, called shares, and securely sends one share to each participant. An access structure describes which subsets of participants are qualified to recover the secret. By a \((t, \ell)\)-threshold access structure, where \(1 \leq t \leq \ell\), we mean that any subset of \(t\) or more participants will be able to recover the secret. In a perfect secret sharing scheme any smaller subset of participants will not be able to gain any information about the secret.

Verifiable secret sharing (VSS) schemes constitute a particularly important class of secret sharing schemes that do not require the dealer to be trusted. The idea of verifiability in secret sharing schemes was introduced by Chor et al. [6]. In a VSS scheme, the shareholders can verify the validity of their shares and thus overcome the problem of dishonest dealers. VSS is known to play important roles in various cryptographic protocols such as the multiparty protocols [4,1], key-escrow cryptosystems [2], and threshold cryptography.
A VSS scheme is called **non-interactive** if the shareholders can verify their share without communicating with each other or the dealer. Efficient non-interactive versions were presented by Feldman [9], where verifiability of the secret is information-theoretic but secrecy relies on computational assumptions, and by Pedersen [19], where verifiability is only guaranteed computationally while secrecy is unconditional.

**(Non-interactive) Publicly Verifiable Secret Sharing:**

The first proposed VSS scheme [6] has the special property that anyone, not only the shareholders, can verify that the shares were correctly distributed. In [24], the property was named **public verifiability** and the VSS schemes with the above property were named publicly verifiable secret sharing schemes (PVSS). Substantial contribution in the field of PVSS came through some of the works in [12, 21, 24]. Recently, bilinear maps (pairings) were used [14, 15] to improve the efficiency of public verifiability in PVSS schemes, and in [8] to build the first construction of a non-interactive PVSS scheme proven secure in the standard model under standard assumption.

1.1 Our Contribution

The Paillier’s public key encryption [18] is well-known and so is its homomorphic property. It is very intuitive that Paillier’s technique could provide efficient PVSS scheme due to its homomorphic property. In this paper we present a PVSS scheme based on Paillier’s encryption scheme (see Section 3). Though our scheme appears more complex than the scheme of Ruiz and Villar, but in efficiency it outperforms the latter with more efficient share distribution process, comparable broadcast bandwidth and public verification process (see Table 1). We formalize the notion of indistinguishability of secrets and prove that our scheme achieves it under the decisional composite residuosity assumption (see Section 4). Our work draws motivation from the well known work on threshold cryptography (threshold public key decryption, threshold signature) based on Paillier’s trapdoor function (see [23, 7, 11, 16, 17]). Some of the tools developed in this research will be useful to this field.

2 Preliminaries

In this section we describe the protocols that form a non-interactive \((t, \ell)\)-threshold publicly verifiable secret sharing (PVSS) scheme, as well as the basic security requirements for such schemes. We also present number-theoretic results required in this paper.

2.1 (Non-interactive) PVSS

We first describe a model for non-interactive PVSS. A distinctive feature of PVSS is that no private channels are assumed between the dealer and the participants and all communication are over (authenticated) public channels using public key encryption. Consequently, the secret will only be hidden computationally.

In a PVSS scheme, a dealer \(D\) wishes to distribute shares of a secret value \(s\) among \(\ell\) participants (players) \(P_1, \ldots, P_\ell\). We consider \((t, \ell)\)-threshold access structure for \(1 \leq t \leq \ell\).
As a common structure for PVSS we consider the following protocols. Note that initialization is done without any interaction between the dealer and the participants. In fact, participants may enter or leave the system dynamically provided that a participant holds a registered public-key. A PVSS scheme is described as follows (see also [21, Section 2]):

- **Initialization:** All the system parameters are generated as part of the initialization. Furthermore, each participant \(P_i\) registers its public-key. The actual set of participants taking part in a run of PVSS scheme must be a subset of the registered participants. We assume w.l.o.g. that participants \(P_1, \ldots, P_\ell\) are the actual participants in the run described below.

- **Distribution:** The protocol consists of two steps:
  1. **Distribution of the shares.** The distribution of a secret \(s\) is performed by the dealer. The dealer first generates the respective shares \(s_i\) for participants \(P_i, 1 \leq i \leq \ell\). For each participant \(P_i\), the dealer publishes the encrypted share accompanied by the “proof of correctness”. Furthermore, the set of proofs commits the dealer to the value of secret \(s\), and it guarantees that the reconstruction protocol will result in the same value \(s\).
  2. **Verification of the shares.** Any party knowing the public keys of the participants may verify the shares. For each participant \(P_i\) a non-interactive verification algorithm can be run on the published proof of correctness by the dealer to verify that a correct encryption of a share for \(P_i\) has been issued by the dealer. Since anyone can verify a share, it can be ruled out that a participant complains while it received a correct share.

- **Reconstruction:** The protocol consists of two steps:
  1. **Decryption of the shares.** The participants decrypt their shares \(s_i\) using their secret keys from the published encrypted shares. It is not required that all participants succeed in doing so, as long as a qualified set of participants is successful. These participants release decryptions of \(s_i\) accompanied by a proof of correctness that shows that the released shares are correct.
  2. **Combining shares to reconstruct the secret.** Reconstruction of the secret \(s\) can be done from the shares of any qualified set of participants.

In non-interactive PVSS schemes it is essential that all commitments can be verified non-interactively. Since any party can verify the output of the dealer, we don’t budget operations for the individual participants to check their own shares. Hence it suffices to have just one public verifier.

2.2 Security Model

Such a scheme must satisfy the following properties.

- **Correctness:** If the dealer and the shareholders act honestly, every qualified subset of shareholders reconstructs the secret during the reconstruction algorithm.

- **Verifiability:** If a dishonest dealer passes the verification step, then there exists a unique value \(s\) such that the honest participants in any qualified subset with at least \(t\) honest participants recover \(s\) as the secret. In the reconstruction protocol, the participants cannot cheat without being detected.

- **Privacy:** The very basic requirement is that, for an honest dealer, the adversary cannot learn any information about the secret at the end of the protocol.
Privacy: We can formally define the above privacy notion, under the classical semantic-security notion \[13\], using a game between an adversary \( A \) and a challenger. The adversary here is a static one, i.e., at the beginning of the game, she is given the secret keys of the corrupted shareholders.

Indistinguishability of Secrets (IND): The security notion is defined via the following game between a challenger and a probabilistic polynomial time (PPT) adversary \( A \). Both the adversary and the challenger are given as input a security parameter \( \mu \).

- **Initialization**: The challenger runs \( \text{Initialization}(\mu) \) to obtain the set of public parameters along with the public keys and the secret keys of all the shareholders. Besides all the public keys, the adversary is also given the secret keys of \( t - 1 \) corrupted shareholders.

- **Find Phase 1**: The adversary chooses a secret \( s \) and the **share distribution oracle** gives her \( \ell \) valid shares of the secret \( s \), along with proofs of validity. This step is repeated as many times as the adversary wishes.

- **Challenge**: The adversary picks two random secrets \( s_0 \) and \( s_1 \) and sends them to the challenger who then randomly chooses a bit \( \tau \in \{0, 1\} \). The challenger runs the distribution algorithm for the secret \( s_\tau \) and sends all the resulting information to \( A \).

- **Find Phase 2**: The adversary repeats Find Phase 1, asking for shares of chosen secrets.

- **Guess**: Finally, the adversary \( A \) outputs a guess bit \( \tau' \in \{0, 1\} \) for \( \tau \) and wins the game if \( \tau' = \tau \).

We define the advantage of this static adversary (SA) \( A \) against a \((t, \ell)\)-threshold PVSS as follows:

\[
\text{Adv}_{\text{SA-IND}}^{\text{PVSS}, A}(\mu) = \left| \Pr[\tau' = \tau] - \frac{1}{2} \right|
\]

The advantage is a function of the security parameter \( \mu \).

**Definition 1** We say that a \((t, \ell)\)-threshold PVSS scheme is SA-IND secure if for all PPT adversaries \( A \), we have that \( \text{Adv}_{\text{SA-IND}}^{\text{PVSS}, A}(\mu) \) is a negligible function in \( \mu \).

Next we discuss the Paillier’s public key encryption scheme. We refer to the Appendix for the related standard notations like \( L \) function, CCRA and DCRA assumptions.

2.3 Paillier’s Public Key Encryption Scheme

Plaintext space: \( \mathbb{Z}_n \); Ciphertext space: \( \mathbb{Z}_{n^2}^\ast \).

- **Key Generation**: Select \( n = pq \), where \( p \) and \( q \) are large primes, and \( p \nmid (q - 1) \) and \( q \nmid (p - 1) \). Set \( \lambda = \text{lcm}(p - 1, q - 1) \). Then select \( g \in \mathbb{Z}_n^\ast \) such that order of \( g \) is a multiple of \( n \) (by checking whether \( \gcd(L(g^\lambda \mod n^2), n) = 1 \)).

**Public Key**: \( (n, g) \) and **Private Key**: \( \lambda \) (or \( p \) and \( q \))

- **Encryption**: Given a plaintext \( M \in \mathbb{Z}_n \), select a random \( r \in \mathbb{Z}_n^\ast \); the ciphertext is given by \( C = g^M r^n \mod n^2 \).

- **Decryption**: Given a ciphertext \( C \in \mathbb{Z}_{n^2}^\ast \), the plaintext is given by

\[
M = \frac{L(C^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n.
\]
Paillier’s Probabilistic Encryption Scheme is one-way if and only if the Computational Composite Residuosity Assumption (CCRA) holds, and semantically secure if and only if the Decisional Composite Residuosity Assumption (DCRA) holds (see Theorems 14 and 15 in [18]).

2.4 Proving Equality of Discrete Logarithms

Let $\mathbb{G}$ be a cyclic group of order $q$. Let $g_1$ be a generator of $\mathbb{G}$ and $g_2$ be any element of $\mathbb{G}$. Chaum and Pedersen [5] described an interactive proof of knowledge of the common logarithm $x$ of $y_1 = g_1^x$ and $y_2 = g_2^x$, where $x$ is a random element from $\{0, 1, \ldots, q - 1\}$.

**Protocol: ProofDLogEq($g_1, g_2, y_1, y_2$)**

- Prover chooses $r \in_R \{0, \ldots, q - 1\}$ and sets $a = (a_1, a_2) = (g_1^r, g_2^r)$. Prover sends $a$ to verifier.
- Verifier chooses $c \in_R \{0, \ldots, q - 1\}$ and sends $c$ to prover.
- Prover computes $b = r + cx$ and sends $b$ to verifier.
- Verifier accepts if and only if $a_1 = g_1^r y_1^c$ and $a_2 = g_2^r y_2^c$.

The above protocol is zero-knowledge if the verifier is an honest one. Thus it is an honest verifier zero-knowledge protocol. The above protocol is a three-move protocol. It is easy to convert an interactive three-move proof into a non-interactive one using the standard method of Fiat-Shamir [10].

Let $h : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ be a collision-resistant hash function. We get a non-interactive proof $(b, c) = \text{ProofDLogEq}(g_1, g_2, y_1, y_2)$ for $y_1 = g_1^x$ and $y_2 = g_2^x$ in the following way. The prover chooses $r \in_R \{0, \ldots, q - 1\}$ and sets $a = (a_1, a_2) = (g_1^r, g_2^r)$. Then she computes the challenge $c = h(g_1||g_2||y_1||y_2||a_1||a_2)$ and sets $b = r + cx$. The verification condition is

$$c = h(g_1||g_2||y_1||y_2||g_1^r y_1^c||g_2^r y_2^c)$$

The verifier doesn’t need to know $a$ to compute the verification condition. If we trust the collision resistance of $h$, we can conclude that $u = v$ from $h(u) = h(v)$. If we convert our proofs of knowledge into non-interactive proofs, honest verifier zero-knowledge is sufficient, because here, the verifier is always honest.

3 Proposed PVSS Scheme

In this section we present a $(t, \ell)$-threshold publicly verifiable secret sharing scheme with $t$ as threshold parameter and $\ell$ denotes the number of participants. Let $\Delta = \ell!$. For any subset $S$ of $t$ elements in $\{0, \ldots, \ell\}$, and for any $i \in \{0, \ldots, \ell\}\setminus S$ and $j \in S$, we can define

$$\lambda_j^i(i) = \Delta \frac{\prod_{k \in S\setminus \{j\}} (i - k)}{\prod_{k \in S\setminus \{j\}} (j - k)} \in \mathbb{Z}$$

These values are derived from the standard Lagrange interpolation formula. They are clearly integers, since the denominator divides $j!(\ell - j)!$ which in turn divides $\ell!$. These numbers were used in [23].
Let \( f(x) = \sum_{i=0}^{t-1} a_i x^i \) be a polynomial over \( \mathbb{Z}_N \) (\( N(>l) \in \mathbb{N} \)) of degree \( t - 1 \), and \( \{(j, f(j))\}_{j \in \mathbb{S}} \) be \( t \) points over \( f(x) \). Then, by Lagrange interpolation, given \( \{(j, f(j))\}_{j \in \mathbb{S}} \) one can reconstruct \( f(i) \) as
\[
\Delta \cdot f(i) \equiv \sum_{j \in \mathbb{S}} \lambda_j^\mathbb{S}(i) f(j) \mod N
\]
Let \( \mathbb{G} \) be a cyclic group of order \( N \) and \( g \) be a generator of \( \mathbb{G} \). From the above identity, we have
\[
g^{\Delta f(i)} = \prod_{j \in \mathbb{S}} g^{\lambda_j^\mathbb{S}(i) f(j)}
\]
By a careful calculation, we observe that it is also possible to reconstruct \( g^{\Delta q_1} \) for \( 0 \leq k \leq t-1 \) from \( (g, \{(j, g^{f(j)})\}_{j \in \mathbb{S}}) \) as
\[
g^{\Delta q_1} = g^{f(k)(0)} = \prod_{j \in \mathbb{S}} g^{(\lambda_j^\mathbb{S}(k)(0) f(j) / k!}
\]
where \( f(k)(0) \) and \( (\lambda_j^\mathbb{S})(k)(0) \) denote the \( k \)th differentiation of the polynomials \( f \) and \( \lambda_j^\mathbb{S} \) and then evaluated at ‘0’. Note that \( k! \) divides \( (\lambda_j^\mathbb{S})(k)(0) \). We use this fact for the security proof in Section 4.2.

Next we propose the following result that will be used in the proposed scheme. An element \( v \in \mathbb{Z}_{n^2}^* \) is a quadratic residue if and only if there exists an element \( w \in \mathbb{Z}_{n^2}^* \) such that \( v \equiv w^2 \mod n^2 \). The set of quadratic residues modulo \( n^2 \) forms a subgroup of \( \mathbb{Z}_{n^2}^* \) and we denote it by \( \mathbb{QR}_{n^2} \). A prime \( p' \) is called a Sophie-Germain prime if \( 2p' + 1 \) is also a prime. In the cryptographic literature, the number \( 2p' + 1 \) (where \( p' \) is a Sophie-Germain prime) is usually called a safe prime. We now let \( n = pq \) be the product of two distinct safe primes \( p \) and \( q \), which we require in our proposed PVSS scheme. In the following lemmas we provide results that will determine an element \( v \in \mathbb{QR}_{n^2} \) to be a generator.

Lemma 1 Let \( n = pq \) be the product of two distinct safe primes \( p \) and \( q \). Then \( \mathbb{QR}_{n^2} \) is a cyclic group. If \( v \) is a generator for \( \mathbb{QR}_{n^2} \) then \( \gcd(v - 1, n) = 1 \).

Proof: Let \( p = 2p' + 1 \) and \( q = 2q' + 1 \), where \( p' \) and \( q' \) are primes. By the Chinese remainder theorem \( \mathbb{QR}_{n^2} \) is isomorphic to \( \mathbb{QR}_{p^2} \times \mathbb{QR}_{q^2} \) given by
\[
v \mapsto (v_{p^2}, v_{q^2}) = (v \mod p^2, v \mod q^2)
\]
As \( \mathbb{QR}_{p^2} \) and \( \mathbb{QR}_{q^2} \) are cyclic and since their orders \( pp' \) and \( qq' \) are relatively prime, \( \mathbb{QR}_{n^2} \) is also cyclic and its order is \( pp'qq' \).

Let \( v' \) be a generator of \( \mathbb{QR}_{p^2} \). Then \( v_{p^2}(v_{p^2}) \) is a generator of \( \mathbb{QR}_{p^2} \) (\( \mathbb{QR}_{q^2} \)). If suppose \( d = \gcd(v - 1, n) \neq 1 \), or equivalently \( d \) is divisible by either \( p \) or \( q \) (or both). Assume w.l.o.g. that \( p \) divides \( d \). Then \( v \equiv 1 \mod p \) and thus \( v^p \equiv 1 \mod p^2 \) implying that \( v_{p^2} \) is not a generator of \( \mathbb{QR}_{p^2} \).

Lemma 2 Let \( v \in \mathbb{QR}_{n^2} \). If \( \gcd(v - 1, n) = 1 \) then \( v \) is a generator with probability \( \frac{\phi(n)}{n} \).

Proof: Let \( o(v_{p^2}) \) and \( o(v_{q^2}) \) be the orders of \( v_{p^2} \) and \( v_{q^2} \) in \( \mathbb{QR}_{p^2} \) and \( \mathbb{QR}_{q^2} \) respectively. The condition \( \gcd(v - 1, n) = 1 \) eliminates the possibility that \( o(v_{p^2}) \模 p \) and \( o(v_{q^2}) \模 q \). Then \( v \) is not a generator of \( \mathbb{QR}_{n^2} \) if it is in the subgroups of order \( pp'q' \) or \( qq'p' \). The total number of elements in these two subgroups is \( pp'q'(p + q - 1) \). Hence the result.

Additionally, by testing \( \gcd(v^{p/q} - 1, n) = 1 \) or not, we can decide upon whether \( v \) is in the subgroups of order \( pp'q' \) or \( qq'p' \), or not. So we have the following lemma.
Lemma 3 Let \( v \in \mathbb{QR}_n \). The element \( v \) is a generator of \( \mathbb{QR}_n \) if and only if \( \gcd(v - 1, n) = 1 \) and \( \gcd(v^m - 1, n) = 1 \).

We now describe our PVSS scheme.

- **Initialization:** All the system parameters are generated as part of the initialization.
  - The dealer generates an integer \( n \), product of two safe primes \( p \) and \( q \), such that \( p = 2p' + 1 \) and \( q = 2q' + 1 \) and \( \gcd(n, \phi(n)) = 1 \). Set \( m = p^2 q' \). Then randomly choose \((a, b) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^* \) and set \( g = (1 + n)^{ab} \mod n^2 \). Randomly choose \( v \in \mathbb{QR}_n^2 \) and check if \( \gcd(v - 1, n) = 1 \) (since the dealer knows \( m \) (factorization of \( n \)), by also checking the validity of \( \gcd(v^m - 1, n) = 1 \), she can guarantee that \( v \) is a generator of \( \mathbb{QR}_n^2 \) by Lemma 3). The dealer finally publishes:
    \[
    \{n, g, v\}
    \]
  - Every participant \( P_i \) randomly selects \((m_i, r_i) \in \mathbb{Z}_n \times \mathbb{Z}_n^* \). They compute and publish:
    \[
    T_i = g^m_i r_i^t \mod n^2 \quad \text{and} \quad W_i = v^{A_{m_i}} \mod n^2
    \]
    The pair \((m_i, r_i)\) is kept secret with the participant \( P_i \).

- **Distribution:** The distribution algorithm consists of two steps:
  1. **Distribution of the shares.** The dealer can distribute a secret \( s \in \mathbb{Z}_m \) among the participants \( P_1, \ldots, P_t \) as follows.
    - Choose \( x \in \mathbb{Z}_m^* \).
    - Compute \( C = g^x \mod n^2 \).
    - Choose \( \beta \in \mathbb{Z}_n^* \).
    - Set \( \theta = \alpha \beta \mod n \).
    - Compute \( m_i = \text{eqn}^{-1}(1) \{H(C_i)^{r_i} \mod n\} \) for \( 1 \leq i \leq t \).
    - Choose a \( t - 1 \) degree polynomial \( f(x) \in \mathbb{Z}_{nm}[x] \):
      \[
      f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}
      \]
      where \( a_0 = \beta m \) and \( a_i \in \mathbb{Z}_{nm} \) for \( 1 \leq i \leq t - 1 \).
    - Compute \( C_i = C^{A_{m_i} - 2\lambda_{m_i}} \), where \( f_i = f(i) \mod nm, 1 \leq i \leq \ell \).
    - Compute \( v^{A_{m_i}}, v^{A_{m_i+1}}, \ldots, v^{A_{m_i+1}} \).
    - Finally publish:
      \[
      \{\theta, C, (C_i)_{1 \leq i \leq \ell}, (v^{A_{m_i}})_{0 \leq i \leq \ell - 1}\}
      \]
    The dealer shows that the encrypted shares are consistent by producing a proof of the unique \( f_i, 1 \leq i \leq \ell \), satisfying:
    \[
    C_i^t = (C^{A_{m_i}})^{f_i + m_i} \quad \text{and} \quad v_i^t W_i = (v^{A_{m_i}})^{f_i + m_i}
    \]
    where \( v_i = v^{A_{m_i}} \text{eqn}^{-1}(1) \prod_{j=0}^{l-1} v^{A_{m_j} f_j} \) for \( 1 \leq i \leq \ell \). The non-interactive proof is the \( \ell \)-fold parallel composition of the protocols for \( \text{ProofDLogEq}(C^{A_{m_i}}, v^{A c_{t}}; v_i \cdot W_i) \), which is carried out as follows. Let \( \delta(n) \) be the bit-length of \( n \) and let \( H \) be a hash function, whose output is an \( \delta_1 \)-bit integer, where \( \delta_1 \) is a secondary security parameter (We have to deal with the fact that we are working in a group \( \mathbb{QR}_n^2 \) whose order is not known to the public verifier and this is dealt with by just working with sufficiently large integers). To construct the proof of correctness for the \( i \)th share, the dealer chooses a random number \( \gamma \in \{0, \ldots, 2^{\delta(n)\delta_1} - 1\} \) and computes
    \[
    v' = (v^{A_{m_i}})^\gamma, \quad C' = (C^{A_{m_i}})^\gamma, \quad e_i = H(v \| C^{A_{m_i}}) \| v_i \| W_i \| C_i^t \| v' \| C')
    \]
The algorithm works in two steps:

1. Decryption of the shares. Using its private key $m_i$, each participant $P_i$ finds her share $C_i' = C_i^{2 \Delta m}$ from $C_i$ by computing $C_i' = C_i \cdot (C_i^{2 \Delta m})^{-1}$. The participants release their shares $C_i'$ accompanied by a proof of correctness that shows the existence of unique $m_i$'s for $1 \leq i \leq t$, satisfying $(C_i C_i'^{-1})^2 = (C_i^{4 \Delta m})^m$ and $W_i = (\psi^A)^m$.

   This is done similar to the public verification of shares in the distribution algorithm by executing the non-interactive version of the protocol:

   \[
   \text{ProofDLogEq}(C_i^{4 \Delta m}, \psi, (C_i'^{-1}, W_i)).
   \]

2. Combining shares to reconstruct the secret. Let there be $t$ participants indexed by the set $S \subseteq \{1, \ldots, t\}$ ($|S| = t$) with the respective valid shares $C_i'$'s. The secret $s$ can be obtained as follows:

\[
L \left( \prod_{i \in S} (C_i')^{2 \Delta m^2(0)} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv L \left( C_i^{4 \Delta \sum_{j=3}^{t} f_i j^{2}(0)} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv L \left( C_i^{2 \Delta f_i} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv L \left( \left( g^s \Delta \right)^{4 \Delta^2 m \beta} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv L \left( \left( 1 + n \right)^{4 \Delta^2 m \beta} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv L \left( \left( 1 + n \right)^{4 \Delta^2 s \beta m \beta} \mod n^2 \right) \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv 4 \Delta^2 s m^2 \beta \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv 4 \Delta^2 s \theta \times \frac{1}{4 \Delta^2 \theta} \mod n
\]

\[
\equiv s \mod n.
\]

The calculations described above, at several places, took help of the equation (6) (with $\Delta = 2m$).

### 3.1 Performance

We compare the efficiency of our scheme (see Table 1) with the scheme of [20], the only known PVSS that is based on Paillier’s encryption and the Schoenmakers scheme [21]. The
Table 1 Comparison Table

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Share Distribution</th>
<th>Broadcast Bandwidth</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruiz and Villar [20]</td>
<td>$\ell(t + 2) + 2t + 1$ exps</td>
<td>$t$ many $\mathbb{Z}_{n^2}$ elts + $2\ell$ many $\mathbb{Z}_n$ elts</td>
<td>$\ell(t + 1)$ exps</td>
</tr>
<tr>
<td>Proposed Scheme</td>
<td>$4\ell + t + 3$ exps</td>
<td>$(\ell + t + 2)$ many $\mathbb{Z}_{n^2}$ elts + $\ell$ many $\mathbb{Z}_n$ elts</td>
<td>$\ell(t + 3)$ exps</td>
</tr>
<tr>
<td>Schoenmakers [21]</td>
<td>$3\ell + t$ exps</td>
<td>$(\ell + t)$ order $q$ group elts + $\ell$ many $\mathbb{Z}_q$ elts</td>
<td>$\ell(t + 3)$ exps</td>
</tr>
</tbody>
</table>

comparison details are given for share distribution, public verification process, and broadcast bandwidth. In share distribution and public verification process, we count only exponentiations. We have not included the count for hash-bits for broadcast bandwidth (essentially one hash output for both the proposed scheme and [21]).

4 Analysis of the Proposed PVSS Scheme

4.1 Verifiability

Now we show that if the dealer passes the verification step, then all the participants in the protocol must behave honestly or will be detected. More precisely, on the one hand, the dealer must be honest in the distribution protocol and, on the other hand, participants must be honest in the reconstruction protocol.

First we observe that a dishonest dealer cannot cheat the participants without being detected in the verification.

**Lemma 4** If the dealer passes the verification step, then all the qualified subsets of honest participants will reconstruct the same secret that the dealer wishes to share.

**Proof:** It follows from the soundness of the Chaum-Pedersen proof and the fact that the $v_i \cdot W_i$’s are obtained from the published commitments $\{v^{\Delta a_j}\}_{0 \leq j \leq t - 1}$ and $\{v^{\Delta m_i}\}_{1 \leq i \leq \ell}$ as

$$v_i \cdot W_i = v^{\Delta f_i} \cdot v^{\Delta m_i} = \prod_{j=0}^{t-1} v^{\Delta a_j} \cdot v^{\Delta m_i}$$

for $1 \leq i \leq \ell$.

Like above, one may also observe that the verifiability of the shares in the reconstruction protocol can also be assured.

4.2 Indistinguishability of Secrets

We now prove that our scheme is secure in the sense of the security model (indistinguishability of secrets) described in Section 2.2. The proof goes by reduction which shows that if an attacker is able to break the semantic security of our PVSS scheme, it can be used to break the semantic security of the Paillier’s encryption scheme described in Section 2.4. The semantic security of Paillier’s scheme is equivalent to Decisional Composite Residuosity Assumption.
The heart of such a reduction proof is a simulator. Given the public key of Paillier’s encryption scheme as input, the simulator plays the security game SA-IND with an attacker against our encryption scheme as input, the simulator plays the security game SA-IND with an attacker. In the Challenge phase, it receives two secrets against our encryption scheme. The simulator then distributes the shares of the secret. Based on the value of the simulator decides whether the Paillier’s encryption it received is of $s_0$ or $s_1$. Intuitively, if the attacker has an advantage in breaking the PVSS scheme, the simulator also has an advantage in breaking the semantic security of the Paillier’s scheme. This leads to an upper bound on the advantage of the attacker in terms of the advantage of the simulator in breaking the semantic security of the Paillier’s encryption scheme.

**Theorem 1** In the random oracle model for $H$, the publicly verifiable secret sharing scheme described in Section 3 is semantically secure assuming the Decisional Composite Residuosity Assumption (DCRA) holds.

**Proof:** Let us assume, there is an attacker $A$ which can break the semantic security of our $(t, \ell)$-threshold PVSS scheme. We construct the simulator $B$ as follows. The simulator first obtains the public key $\{n, g\}$ of the Paillier’s encryption scheme. It plays the security game SA-IND with $A$. The simulator $B$ sets up the initialization phase for the $(t, \ell)$-threshold PVSS as follows.

- First randomly chooses $(c, d) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$ and sets $h = g^c d^n \mod n^2$.
- Picks at random an element $\alpha$ of $\mathbb{Z}_n^*$ and sets $v = h^{2\alpha} \mod n^2$.
- Selects randomly $m_1, \ldots, m_t \in \mathbb{Z}_n$ and sets $W_i = v^{\alpha m_i} \mod n^2$.
- Selects randomly $r_1, \ldots, r_\ell \in \mathbb{Z}_n^*$ and sets $T_i = h^{m_i r_i} \mod n^2$.
- Finally $B$ gives $A$ the public parameters $\{n, h, v\}$.
- It then publishes $T_i$ and $W_i$, $1 \leq i \leq \ell$, as the public keys of the $\ell$ participants.

We now describe how to feed the attacker $A$ of the $(t, \ell)$-threshold PVSS scheme in order to make a semantic attacker. First $B$ receives the list of $t - 1$ corrupted participants from $A$. Without loss of generality, we assume that the first $t - 1$ participants $P_1, \ldots, P_{t-1}$ are corrupted. Then $B$ gives the private keys $\{(m_i, r_i)\}_{1 \leq i \leq t-1}$ of $P_1, \ldots, P_{t-1}$ to $A$.

In Find Phase 1, $A$ chooses a secret $s$ and sends it to $B$. Now $B$ should be able to compute the shares and verification information corresponding to the secret $s$ and it is done as follows. The simulator $B$ first chooses $x \in \mathbb{Z}_n$ and computes $C = h^x \mod n^2$. The shares and the verification information are computed as given below.

- Select at random $t - 1$ values $f_1, \ldots, f_{t-1}$ in the range $\{0, \ldots, \lfloor n^2 \rfloor \}$.
- Choose $\theta \in \mathbb{Z}_n$.
- The shares of the corrupted players are correctly computed using the $f_i$’s:

$$C_i = C^{2A_i + 2A_i} \mod n^2, 1 \leq i \leq t - 1.$$
The shares for the uncorrupted participants are obtained by interpolation are
\[ C_i \equiv (1 + 2\alpha \theta n)^{\lambda_i} \times \prod_{j \in S \setminus \{0\}} c^{2^j \lambda_j} \times C^{2^\Delta m} \mod n^2 \]  
(3) 
where \( S = \{0, 1, \ldots, t - 1\} \) and \( t \leq i \leq \ell \).

- The verification keys are obtained as
\[ v^{\lambda_j} \equiv (1 + 2\alpha \theta n)^{\lambda_j} \times \prod_{j \in S \setminus \{0\}} v^{f_j} \lambda_j^{\lambda_j} \mod n^2 \]  
(4) 
for \( 0 \leq k \leq t - 1 \) and \( S = \{0, 1, \ldots, t - 1\} \).

Then one can compute \( v_i = v^{\lambda_i} \mod n^2 \) for \( 1 \leq i \leq t - 1 \), and for \( t \leq i \leq \ell \)
\[ v_i = (1 + 2\alpha \theta n)^{\lambda_i} \times \prod_{j \in S \setminus \{0\}} v^{f_j} \lambda_j^{\lambda_j} \mod n^2 \]  
(5) 
where \( S = \{0, 1, \ldots, t - 1\} \).

**Indistinguishability of the distribution:**

For \( g \in \mathbb{Z}_n^\times \), there exists \((a, h) \in \mathbb{Z}_n \times \mathbb{Z}_n^\times\) such that \( g = (1 + n)^a b^h \mod n^2 \). Also observe that \( h = g^d \mod n^2 \) is uniformly distributed in the set \( G \) of elements of order multiple of \( n \) (as the order of \( g \) is also a multiple of \( n \)). We need to perform such a modification of \( g \) because we choose \( v \) as an even power of \( h \) (thus making \( v \) a square) and we want \( v \) to generate the subgroup of quadratic residues \( QR_n^2 \). We also observe that \( v \) is uniformly distributed in \( QR_n^2 \). Furthermore, \( v \) is a generator of \( QR_n^2 \) with overwhelming probability (by checking gcd as in Lemma 2).

The simulator chooses the points \( f_1, \ldots, f_{t-1} \) of the corrupted participants; one requirement is that \( f_i \) should be in the interval \( \{0, \ldots, nm\} \). But, since \( m \) is not known, we select \( f_i \in \{0, \ldots, \left\lfloor \frac{n^2}{n} \right\rfloor \} \). The statistical distance between the uniform distribution on \( \{0, \ldots, \left\lfloor \frac{n^2}{n} \right\rfloor - 1\} \) and the uniform distribution on \( \{0, \ldots, nm - 1\} \) is \( O\left(\frac{1}{\sqrt{n}}\right)\), thus the adversary cannot distinguish real and simulated polynomial values of the corrupted participants.

In the simulation we choose \( \theta \in R \mathbb{Z}_n^\times \). Then an element \( \bar{\beta} \in \mathbb{Z}_n^\times \) can be so found as to satisfy \( \bar{\theta} \equiv acm \beta \mod n \). We now show that \( 1 + 2\alpha \theta n \equiv v^{m \beta} \mod n^2 \):
\[
1 + 2\alpha \theta n \equiv (1 + n)^{2\alpha} \mod n^2 \\
\equiv (1 + n)^{2\alpha cm \beta} \mod n^2 \\
\equiv ((1 + n)^{2\alpha} b^n)^{2\alpha cm \beta} \mod n^2 \\
\equiv g^{2\alpha cm \beta} \mod n^2 \\
\equiv (g^d)^{2\alpha cm \beta} \mod n^2 \\
\equiv h^{2\alpha cm \beta} \mod n^2 \\
\equiv (h^{2\alpha} v)^{m \beta} \mod n^2 \equiv v^{m \beta} \mod n^2
\]

Now let \( f(x) = \sum_{k=0}^{t-1} a_k x^k \) be the polynomial over \( \mathbb{Z}_{nm} \) such that
\[ a_0 = f_0 = m \beta \text{ and } f(i) = f_i \text{ for } 1 \leq i \leq t - 1 \]
Thus \( v^{A_{ik}} \) for \( 0 \leq k \leq t - 1 \) (with \( S = \{0,1,...,t-1\} \)) in the equation (4) are obtained correctly as

\[
(1 + 2\alpha \theta n)^{\Delta^k} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(0)/k_1)} \mod n^2
\]

\[
\equiv (v^{m^2})^{\Delta^k} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(0)/k_1)} \mod n^2
\]

\[
\equiv v^{f_0(\Delta^k(0)/k_1)} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(0)/k_1)} \mod n^2
\]

\[
\equiv v^{A_{i(k)(0)}} \equiv v^{A_{ik}}
\]

Of course we are not able to find the missing \( f_i \), \( t \leq i \leq \ell \), however we do not need them and still we can compute \( v_i \) for \( t \leq i \leq \ell \) in the equation (5) as

\[
v_i \equiv (1 + 2\alpha \theta n)^{\Delta^k} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(i))} \mod n^2
\]

\[
\equiv (v^{m^2})^{\Delta^k} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(i))} \mod n^2
\]

\[
\equiv v^{f_0(\Delta^k(i))} \times \prod_{j \in S \setminus \{0\}} v^{f_j(\Delta^k(i))} \mod n^2
\]

\[
\equiv v^{\Delta^k(f_i)} \mod n^2
\]

Similarly like \( v_i \)'s, \( B \) can compute \( C_i \) for \( t \leq i \leq \ell \) in the equation (3) using \( C_1,...,C_{t-1} \) and \( C^{2m^2} \equiv C^{2m^2} \mod n^2 \). Now \( C^{2m^2} \) can be computed without any knowledge of the private information \( m \) as we notice

\[
1 + 2\epsilon \theta n \equiv (1 + n)^{2\epsilon \theta} \mod n^2
\]

\[
\equiv (1 + n)^{2\alpha \epsilon \theta m^2} \mod n^2
\]

\[
\equiv ((1 + n)^{2\epsilon \theta})^{2\alpha \epsilon \theta m^2} \mod n^2
\]

\[
\equiv g^{2\alpha \epsilon \theta m^2} \mod n^2
\]

\[
\equiv (g^e)^{2\epsilon \theta m^2} \mod n^2
\]

\[
\equiv h^{2\epsilon \theta m^2} \mod n^2
\]

\[
\equiv (h^e)^{2\epsilon \theta m^2} \mod n^2
\]

\[
\equiv C^{2\epsilon \theta m^2}
\]

Finally with regard to the “public-verifiability of shares”, the simulator can invoke the random oracle model for the hash function \( H \). We see that the simulator is in complete charge of the random oracle. Whenever \( A \) makes a query to the random oracle, if the oracle has not been previously defined at the given input, the simulator defines it to be a random value, and in any case returns the value to the adversary. Now that the simulator is supposed to generate a proof of correctness for given \( C, C_1,...,C_{t-1}, v_1,...,v_t \), it chooses \( e \in_R \{0,...,2^{6n+2k} - 1\} \) and \( z \in_R \{0,...,2^{6n} - 1\} \) at random, and defines the value of the random oracle at \( (\nu^A, C^{2A}, v_i \cdot W_i, C_i^2, (\nu^A)^z \cdot (v_i \cdot W_i)^z \cdot (C^{2A})^z \cdot (C_i)^z \cdot 2^z) \) to be \( e \) for \( 1 \leq i \leq \ell \). The simulator
is free to do so as with all but negligible probability, it has not defined the random oracle at this point before. The simulator $B$ then outputs $(z, e)$ as the proof of correctness for $C, C_1, \ldots, C_l, v_1, \ldots, v_l$.

Now in Challenge Phase $A$ chooses and outputs two secrets $s_0$ and $s_1$. The simulator outputs those two secrets as messages to the challenger for the Paillier’s encryption scheme. Then the challenger chooses a random bit $\tau \in \{0, 1\}$ and sends an encryption $C^* \equiv x_\tau$ to the simulator. The simulator then computes $\hat{C} \equiv (C^*)^\tau \mod n^2$ and sends $\hat{C}$ to the attacker $A$. Along with $\hat{C}$, the simulator will also have to send the encrypted shares $\{\hat{C}_1\}_{1 \leq i \leq l}$ and $t$ many verification exponents. Like in the Find Phase 1, a random $\theta^* \in Z_n^*$ will give $\nu^{\theta^*}$ (write $\theta^* \equiv cm \beta^* \mod n$), but to compute $C^{2m \beta^*}$ we will no longer use $1 + 2x_\tau \theta^* n$ (as it requires the knowledge of $x_\tau$). Instead we observe that $\hat{C}^{2m \beta^*} = \hat{C}^{\theta^*}$, as

$$(C^*)^{2\theta^*} = (C^*)^{2cm \beta^*} = ((C^*)^{cm})^{2 \beta^*} = \hat{C}^{2m \beta^*}$$

Thus $\hat{C}^{2m \beta^*}$ will be used to compute all the encrypted shares. Also observe that $\hat{C}$ is a valid encryption as $\hat{C} \equiv (C^*)^\tau = (g^{x_\tau r^\tau})^\tau = g^{x_\tau r^\tau} = h^\tau (d^{-x_\tau r^\tau})^\theta$.

The Find Phase 2 is similar to the Find Phase 1. Finally in the Guess Phase, $A$ answers a bit $\tau'$ which is returned by the simulator in the Guess Phase of the game against the challenger for the Paillier’s encryption scheme. □

Acknowledgements A portion of the work has been done when the first two authors were at C R Rao AIMSCS, Hyderabad, India. Financial support for this research was provided in part by Alberta Innovates - Technology Futures, in the Province of Alberta in Canada.

References


A Composite Residuosity Class Assumptions

Let \( n = pq \) be the product of two distinct large primes \( p \) and \( q \). We will consider the group \( \mathbb{Z}_n^* \) under multiplication. We have \( |\mathbb{Z}_n^*| = n\phi(n) \), where \( \phi(n) = (p-1)(q-1) \) is the Euler’s totient function. Unless otherwise mentioned let \( n = pq \) be such that \( \gcd(\phi(n), n) = 1 \), i.e., \( p \nmid (q-1) \) and \( q \nmid (p-1) \). Let \( \lambda(n) = \text{lcm}(p-1, q-1) \), where \( \lambda(n) \) is called Carmichael’s function. For simplicity we write \( \lambda \) instead of \( \lambda(n) \) here onwards.

**Definition 2** An element \( x \in \mathbb{Z}_n^* \) is said to be an \( n \)-th residue modulo \( n^2 \) if there exists \( y \in \mathbb{Z}_n^* \) such that

\[
x \equiv y^\lambda \mod n^2
\]

We have the following results from [18]. Let \( n, \lambda \) be as described above. Then for any \( w \in \mathbb{Z}_n^* \), we have

\[
w^\lambda \equiv 1 \mod n \quad \text{and} \quad w^{n\lambda} \equiv 1 \mod n^2
\]

(6)

Every \( n \)-th residue in \( \mathbb{Z}_n^* \) has exactly \( n \) different \( n \)-th roots in \( \mathbb{Z}_n^* \). The \( n \)-th residues in \( \mathbb{Z}_n^* \) form a subgroup and its order is \( \phi(n) \). The elements in

\[S_n = \{ u = 1 + vn \mid 0 \leq v \leq n-1 \}\]

are the \( n \)-th roots of unity in \( \mathbb{Z}_n^* \). The set \( S_n \) is a cyclic subgroup of \( \mathbb{Z}_n^* \). If \( y \) is an \( n \)-th root of an \( n \)-th residue \( x \in \mathbb{Z}_n^* \), then the \( n \)-th roots of \( x \) are given by \( yS_n = \{ y(1 + vn) \mod n^2 \mid 0 \leq v \leq n-1 \} \).

The problem of deciding \( n \)-th residuosity, i.e., distinguishing \( n \)-th residues from non-\( n \)-th residues will be denoted by \( \text{CR}[n] \). The \( \text{CR}[n] \) problem is believed to be computationally hard. Accordingly, we will assume that: There exists no polynomial time distinguisher for \( n \)-th residues modulo \( n^2 \), i.e., \( \text{CR}[n] \) is intractable. This intractability hypothesis will be referred to as the **Decisional Composite Residuosity Assumption (DCRA)**.

Now we define \( n \)-th residuosity class for elements in \( \mathbb{Z}_n^* \). Let \( G \) denote the set of all elements \( g \in \mathbb{Z}_n^* \), where the order of \( g \) is a multiple of \( n \). Let \( g \in G \) and \( \mathcal{E}_g \) be the function defined by

\[
\mathcal{E}_g : \mathbb{Z}_n \times \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*
\]

\[(x, y) \rightarrow g^x y^\lambda \mod n^2\]

We can show that \( \mathcal{E}_g \) is bijective (see [18, Lemma 3]), and thus the following definition is valid.
Definition 3 For \( w \in \mathbb{Z}_{n^2}^* \) and \( g \in G \), we call \( n \)-th residuosity class of \( w \) with respect to \( g \) the unique integer \( x \in \mathbb{Z}_n \) for which there exists \( y \in \mathbb{Z}_n^* \) such that
\[
E_g(x, y) = g^{xy^n} = w
\]
The \( n \)-th residuosity class of \( w \) with respect to \( g \) is denoted by \([w]_g(=x)\).

The \( n \)-th Residuosity Class Problem of base \( g \), denoted Class\([n]; g\), is defined as the problem of computing the class function in base \( g \): for a given \( w \in \mathbb{Z}_{n^2}^* \), compute \([w]_g\) from \( w \).

From the following discussion one can see that the knowledge of \( \lambda \) (or factorization of \( n \)) is sufficient to solve Class\([n]\) problem. For this purpose, we define a function \( L \) on \( S_n = \{ u = 1 + vn | 0 \leq v \leq n - 1 \} \) by
\[
L(u) = \frac{u - 1}{n}
\]
We can easily see that \( L(u) = \left[u\right]_{1+n} \). We also have \( L(w^k \mod n^2) = \lambda \left[w\right]_{1+n} \) for any \( w \in \mathbb{Z}_{n^2}^* \) (see [18, Lemma 10]). Note that \( w^k \mod n^2 \) (mod \( n^2 \)) \( \in S_n \) by (6). In fact, we can compute \([w]_g\) for a given \( w \in \mathbb{Z}_{n^2}^* \) and \( g \in G \) by the following formula.
\[
[w]_g \equiv \frac{L(w^k \mod n^2)}{L(g^k \mod n^2)} \mod n (7)
\]
Computing \([w]_g\) without the knowledge of \( \lambda \) (or \( p \) and \( q \)) is assumed to be hard. In particular, we have the following computational hierarchy of the problems.

Let D-Class\([n]\) be the decisional problem associated to Class\([n]\), i.e., given \( w \in \mathbb{Z}_{n^2}^* \) and \( x \in \mathbb{Z}_n \), decide whether \( x = [w]_g \) or not. We denote by RSA\([n, e]\) the well-known problem of extracting \( e \)-th roots modulo \( n \), where \( n = pq \) is of unknown factorization and by Fact\([n]\) the problem of factorizing \( n \). Then we have (see [18, p. 228])
\[
\text{CR}[n] \equiv \text{D-Class}[n] \leftarrow \text{Class}[n] \leftarrow \text{RSA}[n, e] \leftarrow \text{Fact}[n]
\]
Quantum algorithm to check Resiliency of a Boolean function (Extended Abstract)

Kaushik Chakraborty · Subhamoy Maitra

Abstract In this paper, for the first time, we present quantum algorithms to check the order of resiliency of a Boolean function. We first show that the Deutsch-Jozsa algorithm can be directly used for this purpose. We also point out how the quadratic improvement in query complexity over the Deutsch-Jozsa algorithm can be obtained using the well known Grover’s algorithm. While the worst case quantum query complexity to check the resiliency order is exponential, we can cleverly devise a strategy so that the number of measurements are polynomial in number of input variables of the Boolean function. We also point out a subset of \( n \)-variable Boolean functions for which the algorithm works in polynomial many steps, i.e., we achieve exponential speed-up over best known classical algorithms.

Keywords: Boolean Functions, Deutsch-Jozsa Algorithm, Grover Algorithm, Measurement, Resiliency.

1 Introduction

After the introduction of Deutsch-Jozsa algorithm [4] in quantum paradigm, several works have been presented in literature to describe strategies that can distinguish Boolean functions of different weights (for example, see [1] and the references therein). Such problems are actually related to studying Walsh spectrum of Boolean functions. From cryptologic viewpoint, the concept of balancedness can be generalized with the idea of resiliency and we will consider this problem here. We will study how efficiently one can check whether a Boolean function is \( m \)-resilient or not.

Before proceeding further, let us introduce basics of Boolean functions. A Boolean function on \( n \) variables may be viewed as a mapping from \( \{0,1\}^n \) into \( \{0,1\} \). We will denote the set of \( n \)-variable Boolean functions as \( B_n \). It is easy
to note that $|\mathcal{B}_n| = 2^{2^n}$. Let us denote the addition operator over $GF(2)$ by $\oplus$. Let $x = (x_1, \ldots, x_n)$ and $\omega = (\omega_1, \ldots, \omega_n)$ both belonging to $\{0, 1\}^n$ and the inner product $x \cdot \omega = x_1\omega_1 \oplus \cdots \oplus x_n\omega_n$. Let $f(x)$ be a Boolean function on $n$ variables. Then the Walsh transform of $f(x)$ is an integer valued function over $\{0, 1\}^n$ which is defined as $W_f(\omega) = \sum_{x \in \{0, 1\}^n} (-1)^{f(x) \oplus x \cdot \omega}$. The fastest known classical algorithm to calculate all the Walsh spectrum values of $f \in \mathcal{B}_n$, i.e., $W_f(\omega)$ at each of the $2^n$ points $\omega$, is of $O(n2^n)$ time complexity. To calculate the Walsh spectrum value at a specific point requires $O(2^n)$ time too in classical domain.

For a binary string $str$, the number of 1’s in the string is called (Hamming) weight of $str$ and denoted as $wt(str)$. In truth table representation, a Boolean function $f \in \mathcal{B}_n$ can be viewed as a binary string of length $2^n$, which is the output column of the truth table. If $wt(f) = 2^{n-1}$, then $f$ is called a balanced function. In terms of Walsh spectrum, $f \in \mathcal{B}_n$ is balanced if and only if $W_f(0, 0, \ldots, 0) = 0$. Following [7], a function $f \in \mathcal{B}_n$ is $m$-resilient iff its Walsh transform satisfies $W_f(\omega) = 0$, for $0 \leq wt(\omega) \leq m$. It is easy to note that a balanced function is actually a 0-resilient function. Thus, informally speaking, the problems related to resiliency will be the generalization of the problems related to balancedness.

Before proceeding further, let us briefly discuss certain algorithms in classical as well as quantum domain. We consider that the Boolean function $f \in \mathcal{B}_n$ is available as an (classical or its quantum counterpart) oracle, i.e., the corresponding output can be obtained efficiently given the input. Given the promise that $f$ is either constant or balanced, to check which one it is, we have Deutsch-Jozsa [4] algorithm to solve it in constant time. Though there is no polynomial time algorithm to solve this problem in classical domain, probabilistic polynomial time algorithms are indeed available to solve this problem efficiently. Given that $W_f(\omega) = 0$ or $\pm 2^n$, the question of “which one it is” can be solved exactly in a similar manner, by considering the function $f(x) \oplus \omega \cdot x$ instead of $f(x)$.

It is well known that checking resiliency of an $n$-variable Boolean function requires exponential time in $n$ in classical domain. In this paper we try to analyse the solution of this problem in quantum paradigm. We note that the traditional Deutsch-Jozsa [4] can be used for this purpose. Further, we try to devise strategies with better efficiency than this using Grover algorithm [5]. It should be noted that Grover algorithm has earlier been used in weight decision problems for Boolean functions [1] and we note that a more involved application of this algorithm can also be exploited in the resiliency checking problem. The most important contribution of our work is that, though the worst case query complexity of our algorithm may be exponential, we need only polynomial many measurements for this purpose. We also identify a sub class of Boolean functions for which our quantum algorithms work with polynomial many queries in $n$. The best known classical algorithm for this sub class requires exponential many steps.

2 Algorithm to check Resiliency

Given $f$ is either constant or balanced, if the corresponding quantum implementation $U_f$ is available, Deutsch-Jozsa [4] provided a quantum algorithm that decides

\footnote{For quantum algorithms, we write “query complexity” instead of “time complexity" as we need to query some oracles, e.g., $U_f, O_{ij}$ as described in Section 2.}
in constant many queries which one it is. The overall idea of the algorithm can be summarized as in Figure 1.

Fig. 1 Quantum circuit to implement Deutsch-Jozsa Algorithm

The step by step description of the Deutsch-Jozsa [4] algorithm can be written as follows.

**Algorithm 1**: The Deutsch-Jozsa algorithm [4].
Proposition 1 Given an \( n \)-variable Boolean function \( f \), \( \mathcal{D}_f |0 \rangle \otimes |n \rangle \) produces a superposition of all states \( z \in \{0, 1\}^n \) with the amplitude \( \frac{W_f(z)}{2^n} \) corresponding to each state \( |z \rangle \).

Consider that we are interested to know whether \( f \in \mathcal{B}_m \) is \( m \)-resilient. Let \( S_m = \{ x \in \{0, 1\}^n \mid \text{wt}(x) \leq m \} \) and \( \overline{S}_m = \{ x \in \{0, 1\}^n \mid \text{wt}(x) > m \} \). Consider the \( n \)-qubit state \( |\Psi \rangle = \sum_{s \in S_m} \frac{W_f(s)}{2^{rn}} |s \rangle + \sum_{s \in \overline{S}_m} \frac{W_f(s)}{2^{rn}} |s \rangle \). For brevity, let us represent \( |\Psi \rangle = a |X \rangle + b |Y \rangle \). That is, \( a^2 = \sum_{s \in S_m} \frac{W_f(s)}{2^{rn}} \) and \( b^2 = \sum_{s \in \overline{S}_m} \frac{W_f(s)}{2^{rn}} \).

Using the Deutsch-Jozsa algorithm, we obtain \( \sum_{z \in \{0, 1\}^n} \frac{W_f(z)}{2^n} |z \rangle \) (before the measurement) and \( a^2 = \sum_{s \in S_m} \frac{W_f(s)}{2^n} \). That is, some state \( s \in S_m \) will appear after the measurement with probability \( a^2 \). Hence, in expected \( O\left(\frac{1}{r} \right) \) iterations, one can observe some \( s \in S_m \) after the measurement and output that \( f \) is not \( m \)-resilient. If \( f \) is indeed \( m \)-resilient, then \( a^2 = 0 \) and thus any state \( s \in S_m \) will never appear at the output. One may note that the minimum absolute value of Walsh spectrum is 2 and thus, we can have a situation that \( f \) is not \( m \)-resilient, but \( a^2 \) is \( O\left(\frac{1}{2^m} \right) \).

In such a case, the algorithm will require exponential many queries to provide the correct result. Thus the resiliency checking algorithm is as follows.

**Algorithm 2**: Resiliency checking using the Deutsch-Jozsa algorithm [4].

| Input: | A Boolean function \( f \) on \( n \) variables is available in the form of the unitary transformation \( U_f \), order of resiliency \( m \) and the number of iteration \( r \). |
| Output: | \( n \)-bit pattern |
| \( 1 \) | \( S_m = \{ x \in \{0, 1\}^n \mid \text{wt}(x) \leq m \} \); |
| \( 2 \) | for \( r \) many times do |
| \( 3 \) | Apply Deutsch-Jozsa algorithm and take the \( n \)-bit output \( u \); |
| \( 4 \) | if \( u \in S_m \) then |
| \( 4.1 \) | Report that the function is not \( m \)-resilient (NO) and terminate; |
| \( 5 \) | Report that the function is \( m \)-resilient (YES); |

**Theorem 1** Let \( c \) be a predefined constant. Algorithm 2 correctly answers \( NO \), but answers \( YES \) with success probability greater than or equal to \( c \), in \( r \) many steps, where \( r \) is \( O\left(\frac{1}{2^m} \right) \) and \( a^2 = \sum_{s \in S_m} \frac{W_f(s)}{2^{rn}} > 0 \).

**Proof** According to Algorithm 2, one can observe that for each iteration, the success probability is \( a^2 \). At \( i \)-th step, the success probability will be \( 1 - (1 - a^2)^i \). So, at \( i = r \) the success probability will become \( 1 - (1 - a^2)^r = c \). Now solving this equation we get \( r \) is \( O\left(\frac{1}{2^m} \right) \).

**Remark 1** Algorithm 2 is written in such a manner that if a function is indeed \( m \)-resilient, then \( a = 0 \) and thus the algorithm will say \( YES \) after executing \( r \) many steps. However, it is known that for nonzero Walsh spectrum values, the minimum is \( \pm 2 \) and thus, \( a^2 \geq \frac{4}{2^rn} \). Hence, after repeating the algorithm \( r \), i.e., \( O\left(2^{2n} \right) \) many
Quantum algorithm to check Resiliency of a Boolean function (Extended Abstract) 5

times, if we don’t observe any binary string \(u \in S_m\) after measurements, then we can conclude that the Boolean function \(f\) is \(m\)-resilient with success probability greater than some predefined constant \(c\). This provides the worst case scenario.

2.1 Improvement using Grover Algorithm

Grover algorithm [5] provides a quadratic speed-up compared to repeated use of Deutsch-Jozsa algorithm and that is the motivation we try out here. Instead of equal superposition \(|\psi\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} |x\rangle\) in Grover algorithm, we will use the state of the form \(|\Psi\rangle = D_f(|0\rangle^{\otimes n}) = \sum_{x \in \{0,1\}^n} \frac{W_f(x)}{2^n} |x\rangle\).

Consider that any \(n\)-qubit state is represented in the computational basis. We want to amplify the amplitude at the points in \(S_m\). This we achieve in a similar manner as in Grover algorithm.

The Grover algorithm requires inversion of phase. Towards this, we will use \(g(x) \in B_n\), different from \(f(x)\). The corresponding operator \(O_g\) inverts the phase of the states \(|x\rangle\) where \(x \in S_m\). That is, we need to change phase for the points having weight less than or equal to \(m\). This can be achieved by choosing the \(n\)-variable Boolean function \(g(x)\) such that \(g(x) = 1\), when \(w(x) \leq m\), and \(g(x) = 0\), otherwise. Thus \(g\) is a symmetric function. A symmetric Boolean function can be efficiently implemented, as described in [3]. The circuit complexity of an \(n\)-variable symmetric Boolean function is \(4.5n + o(n)\). It is known that given a classical circuit \(g\), a quantum circuit of comparable efficiency can be implemented. Thus, we will consider that for a symmetric function \(g\), the quantum circuit \(O_g\) can be efficiently implemented using \(O(n)\) circuit complexity.

Now let us consider the operator \(G_t = \left[|(2|\Psi\rangle - I)O_g\right]^t\) on \(|\Psi\rangle\) to get \(|\Psi_t\rangle\).

The idea presented in the following result is similar to amplitude amplification for constructing Dicke states as presented in [2]. However, we present the proof for better understanding.

**Theorem 2** Let \(|\Psi\rangle = \sum_{s \in S_m} \frac{W_f(s)}{2^n} |s\rangle + \sum_{s \in S_m} \frac{W_f(s)}{2^n} |s\rangle = a|X\rangle + b|Y\rangle\), where \(a = \sin \theta\), \(b = \cos \theta\). The application of \([(2|\Psi\rangle - I)O_g]^{t+1}\) operator on \(|\Psi\rangle\) produces \(|\Psi_t\rangle\), in which the probability amplitude of |X\rangle is \(\sin(2t+1)\theta\).

**Proof** For \(t = 1\), one can check that

\[|\Psi_1\rangle = [(2|\Psi\rangle - I)O_g]|\Psi\rangle = [(2|\Psi\rangle - I)O_g]|\Psi\rangle - O_g|\Psi\rangle.\]

Now substituting the values of \(a, b\) we get that \(|\Psi_1\rangle = \sin 3\theta |X\rangle + \cos 3\theta |Y\rangle\).

Now we will use induction. Let the application of \([(2|\Psi\rangle - I)O_g]^{k+1}\) operator on \(|\Psi\rangle\) updates the probability amplitude of \(|X\rangle\) as \(\sin(2k\theta + \theta)\), for \(t = k\). From the assumption we have \([(2|\Psi\rangle - I)O_g]^k|\Psi\rangle = \sin(\theta + 2k\theta)|X\rangle + \cos(\theta + 2k\theta)|Y\rangle\). Now for \(t = k + 1\), it can be checked that

\[[(2|\Psi\rangle - I)O_g]^{(k+1)}|\Psi\rangle = \sin(\theta + 2(k+1)\theta)|X\rangle + \cos(\theta + 2(k+1)\theta)|Y\rangle.\]

Thus, the proof. \(\square\)
After the Deutsch-Jozsa algorithm we obtain \( \sum_{z \in \{0,1\}^n} \frac{W_f(z)}{2^n} |z\rangle \) (before the measurement) with \( a^2 = \sum_{s \in S_{m}} \frac{W_f(s)}{2^n} \) and \( b^2 = \sum_{s \in \overline{S}_m} \frac{W_f(s)}{2^n} \). Thus, we have \( \sin \theta = a \).

For large \( n \), one can approximate it as \( \theta = a \) and hence we need \( t \) iterations of Grover like strategy such that \( (2t+1)\theta \geq \sin^{-1} c \), where \( c \) is a predefined constant. Thus, here we need an expected \( O(\frac{1}{\theta}) \) iterations, compared to \( O(\frac{1}{a^2}) \) iterations using the Deutsch-Jozsa algorithm only.

---

**Input:** A Boolean function \( f \) on \( n \) variables is available in the form of the unitary transformation \( U_f \), order of resiliency \( m \) and the number of iteration \( r \) and a series of positive integers \( t_i \), \( 1 \leq i \leq r \) related to number of Grover iteration

**Output:** \( n \)-bit pattern

1. \( S_m = \{ x \in \{0,1\}^n | wt(x) \leq m \} \);
2. for \( i = 1 \) to \( r \) do
   3. Apply Deutsch-Jozsa algorithm till the step before measurement to obtain \( |\Psi \rangle = \sum_{s \in S_m} \frac{W_f(s)}{2^n} |s\rangle + \sum_{s \in \overline{S}_m} \frac{W_f(s)}{2^n} |s\rangle \);
   4. By applying Grover iteration, obtain \( |\Psi_{t_i} \rangle = (|2\langle \Psi|\langle \Psi | - I\rangle O_f |\Psi \rangle)^i |\Psi \rangle \);
   5. Measure \( |\Psi_{t_i} \rangle \) in computational basis to obtain \( n \)-bit string \( u \);
   6. if \( u \in S_m \) then
      7. Report that the function is not \( m \)-resilient (NO) and terminate;
   end
7. Report that the function is \( m \)-resilient (YES);

Algorithm 3: Resiliency checking using the Grover algorithm [4].

One important issue here is that any estimate of \( a \) may not be known and thus, estimating \( t_r \) could be challenging. Given that \( t_r \) can be estimated, after application of Grover’s algorithm, we will obtain a state \( \sum_{s \in S_m} a'_s |s\rangle + \sum_{s \in \overline{S}_m} b'_s |s\rangle = a'|X\rangle + b'|Y\rangle \), where \( (a')^2 \) is very close to 1. Using this (Grover algorithm followed by Deutsch-Jozsa algorithm), we get a quadratic speed-up over just using Deutsch-Jozsa algorithm.

It is natural to use Grover algorithm for amplitude amplification and thus obtaining quadratic speed-up. However, in the known applications (e.g., search), the number of target states for which the amplitude is increased are not large. That guarantees the efficient implementation of the phase reversal circuit. In this case, the situation is different as we need to amplify the amplitude at \( \sum_{i=0}^{m} \binom{n}{i} \) many points of weight \( \leq m \) and this could be exponential. Thus, it is an important question whether the phase reversal can be implemented efficiently. In this case, this can be achieved as the phase reversal can be implemented with symmetric functions, the implementation of which is efficient [3].

2.2 Deciding the numbers of Grover iteration

Now let us explicitly describe how one can decide the values of \( t_i \) for \( 1 \leq i \leq r \). As given in Algorithm 3, we have \( |\Psi \rangle = \sum_{s \in S_m} \frac{W_f(s)}{2^n} |s\rangle + \sum_{s \in \overline{S}_m} \frac{W_f(s)}{2^n} |s\rangle = a'|X\rangle + b'|Y\rangle \), where \( a = \sin \theta \), \( b = \cos \theta \). Our motivation is to observe some state
Similarly, we have

Thus, from (1), (4), we get,

\[ (2t_i + 1)\alpha_i = \pi - \theta_c, \quad (2t_i + 1)\alpha_{i+1} = \theta_c. \]

Similarly, we have

\[ (2t_{i-1} + 1)\alpha_{i-1} = \pi - \theta_c, \quad (2t_{i-1} + 1)\alpha_i = \theta_c. \]

Thus, from (1), (4), we get,

\[
\frac{2t_i + 1}{2t_{i-1} + 1} = \frac{\pi - \theta_c}{\theta_c}.
\]

Taking the initial condition \( t_1 = 0 \) and by solving the above recurrence relation, we get,

\[
(2t_i + 1) = \frac{(\pi - \theta_c)(i-1)}{\theta_c^{i-1}} \tag{6}
\]

Now the question is how many times we have to continue this process or what should be the value of \( r \). To answer this question we have to consider in the worst case given the value of \( \sin \theta \). Let \( \sin \theta = a \). From Theorem 2 we know that to ensure \( \sin \theta \) to be close to 1, the maximum value among \( t_i \)'s, i.e., \( t_r \) according to our technique, should be taken as \( O\left(\frac{1}{a}\right) \). So, \( 2t_r + 1 \approx \frac{1}{a} \) and we can write \( r \approx \log_{\frac{1}{\theta_c}} \left(\frac{1}{a}\right) \), i.e., \( r \) is \( O(\log \frac{1}{a}) \). Thus, we have the following result.

**Theorem 3** Let \( c \) be a predefined constant. Algorithm 3 correctly answers NO, but answers YES with success probability greater than or equal to \( c \), i.e., \( O(\log \frac{1}{a}) \) many steps and the number of times the Grover operator is executed is \( O\left(\frac{1}{a}\right) \) where \( a^2 = \sum_{s \in S_m} W_f(s) \).

**Proof** How we estimate \( r \) is explained above. In Algorithm 3, at the \( i \)-th step we apply the operator \( [(2\psi\langle\psi| - I)O_3] \), \( t_i \) times. Here \( i \) varies from 1 to \( r \). So, the total number of times the Grover operator is applied is \( T = \sum_{i=1}^{r} t_i \). From (6), we can substitute the value of \( t_i \) and get \( T = \frac{1}{2} \sum_{i=1}^{r} \left( \frac{(\pi - \theta_c)(i-1)}{\theta_c^{i-1}} \right) - 1 \). By solving this equation and also substituting the value of \( r \) we get,

\[
T \approx \frac{1}{2} \left[ \frac{1/a - 1}{(\pi - \theta_c)/\theta_c - 1} - \frac{1}{2} \left\{ \log_{\frac{1}{\theta_c}} \left(\frac{1}{a}\right) \right\} \log_{\frac{1}{\theta_c}} \left(\frac{1}{a}\right) + 1 \right]. \tag{7}
\]
So, the number of times the Grover operator is executed is $O(\frac{1}{a})$. □

**Remark 2** Similar to Remark 1, for Algorithm 3 we need to consider the case when the function is $m$-resilient, i.e., $a = 0$. In this case $r$ will be $O(\log 2^n)$, i.e., $O(n)$ and $t_r$ will be $O(2^n)$, that provides the worst case scenario.

**Remark 3** We like to point out that the number of measurements in both Algorithm 2 and Algorithm 3 are $r$. In case of Algorithm 2, $r$ is $O(\frac{1}{a^2})$ and can be exponential in $n$ worst case. However, for Algorithm 3, $r$ is $O(\log \frac{1}{a})$, which is polynomial in $n$ in worst case. For Algorithm 2, the number of queries using the Deutsch-Jozsa operator is $r = O(\frac{1}{a^2})$ and for Algorithm 3, the number of queries using the Grover operator is $T = O(\frac{1}{a})$ and both of them could be exponential in the worst case. In summary,

- in terms of number of queries, Algorithm 3 provides quadratic improvement over Algorithm 2, though both can be exponential in worst case;
- in terms of number of measurements, Algorithm 3 requires polynomial many measurements in worst case, while Algorithm 2 requires exponential many.

### 3 Checking $m$-resiliency among functions with three valued Walsh spectrum

Form the analysis in the previous section, we note that the Deutsch-Jozsa algorithm or the Deutsch-Jozsa algorithm (without measurement) followed by the Grover algorithm can be used to check whether a Boolean function is $m$-resilient or not. It is very clear that the second strategy provides a quadratic speed-up over the first one. It is also evident that the quantum algorithms, in worst case, may take exponential queries in $n$. Thus, it would be interesting to consider a class of Boolean functions for which the problem can be solved in polynomial many queries in $n$ in quantum paradigm.

In [9–11], several characterizations and constructions of resilient functions have been presented. In particular, it has been pointed out in [9] that the Walsh spectrum values of any $m$-resilient function will be divisible by $2^{m+2}$. In this direction, we will concentrate on Boolean functions with Walsh spectrum values multiple of $2^{m+2}$. Let us define

$$A_n = \{f \in B_n | W_f(\omega) \equiv 0 \mod 2^{m+2}\}.$$  

In this case, if the function is not $m$-resilient, then $a \geq 2^{m+2}$ and thus, the query complexity of checking resiliency is $O(2^{n-m-2})$ using Algorithm 3. In case, $m \geq n - O(poly(\log n))$, it is clear that the query complexity of checking resiliency is $O(poly(n))$.

We do not know of any classical algorithm that can efficiently decide whether a function $f \in A_n$ is $m$-resilient. Thus, we get an exponential speed-up in this case using quantum algorithm over classical ones.
4 Conclusion

For the first time, we study the problem of checking resiliency of a Boolean function in quantum paradigm. We try to obtain algorithms where the input is an $n$-variable, $m$-resilient Boolean function and the algorithm should output YES if the function is $m$-resilient and NO if it is not. Our algorithm provides the NO answer correctly, while the YES answer with probability greater than some predefined constant. We use the well known Deutsch-Jozsa and Grover algorithms for the purpose. Algorithm 3 shows that it requires exponential many queries but polynomial many measurements in $n$ in the worst case. We also identify a subclass of Boolean functions for which we require polynomial many queries as well as polynomial many measurements in the worst case. For such a class no efficient classical algorithm is known. A more elaborate study for such sub-classes will be presented in the full version of this paper.

References

Construction X for quantum error-correcting codes

Petr Lisoněk · Vijaykumar Singh

Abstract

Construction X is known from the theory of classical error control codes. We present a variant of this construction that produces stabilizer quantum error control codes from arbitrary linear codes. Our construction does not require the classical linear code that is used as an ingredient to satisfy the dual containment condition. We prove lower bounds on the minimum distance of quantum codes obtained from our construction. We give examples of record breaking quantum codes produced from our construction.

Keywords

Quantum error-correcting code · Linear code

Mathematics Subject Classification (2000) 94B05 · 94B15

1 Quantum stabilizer codes

Let \( F_q \) denote the finite field with \( q \) elements and \( F_q^* := F_q \setminus \{0\} \). For \( a \in F_4 \) let \( \pi = a^2 \). For \( x, y \in F_4^n \) let \( \langle x, y \rangle = \sum_{i=1}^n x_i y_i \) be their Hermitian inner product. For a subspace \( C \subset F_4^n \) let \( C^{\perp_h} := \{ u \in F_4^n : \langle u, x \rangle = 0, \forall x \in C \} \) be the Hermitian dual of \( C \). For two subspaces \( U, V \subset F_4^n \) denote \( U + V := \{ u + v : u \in U, v \in V \} \) and \( U \cap V = \{0\} \) and we will say that \( U \oplus V \) is the direct sum of \( U \) and \( V \).

For \( x \in F_4^n \) let \( \text{wt}(x) \) denote the Hamming weight (or just weight) of \( x \), that is, the number of non-zero coordinates of \( x \). By an \([n,k]_q\) linear code we mean a \( k \)-dimensional subspace of \( F_q^n \). By an \([n,k,d]_q\) linear code we mean an \([n,k]_q\) linear code \( C \) such that the minimum weight of any non-zero vector in \( C \) is \( d \). For a linear code \( C \) we denote by \( \text{wt}(C) \) the minimum weight of any non-zero vector.

Research reported in this paper was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Collaborative Research Group “Mathematics of Quantum Information” of the Pacific Institute for the Mathematical Sciences (PIMS).

P. Lisoněk · V. Singh
Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada V5A 1S6
E-mail: plisonek@sfu.ca, vijay.k.1@gmail.com
in $C$. As is well known, $\text{wt}(C)$ also equals the minimum Hamming distance of any two distinct codewords in $C$.

A quantum error-correcting code (or just quantum code) is a code that protects quantum information from corruption by noise (decoherence) on the quantum channel in a way that is similar to how classical error-correcting codes protect information on the classical channel. We recommend [3] as an excellent reference on binary quantum error-correcting codes. In accordance with literature we denote by $[[n, k, d]]$ the parameters of a binary quantum code that encodes $k$ logical qubits into $n$ physical qubits and has minimum distance $d$. (Sometimes such parameters are denoted $[[n, k, d]]_2$ to emphasize that the quantum code is binary; however as we only deal with binary quantum codes in this paper, we will omit the subscript 2 throughout.) The minimum distance of a quantum code $C$ determines the error detection/correction capabilities of $C$ in a way that is similar to how the minimum distance determines such capabilities of classical codes: For fixed $n$ and $k$, the higher $d$ is, the more error control the code achieves.

An important class of quantum codes are stabilizer quantum codes. Binary stabilizer quantum codes are equivalent to quaternary additive codes, that is, additive subgroups of $\mathbb{F}_4^n$ [3]. If we further restrict our attention to linear subspaces of $\mathbb{F}_4^n$, then the following theorem expresses the parameters of the quantum code that can be constructed from a classical linear, dual containing code.

**Theorem 1** [3, Theorem 2] [1, Theorem 17.24] Given a linear $[n, k, d]_4$ code $C$ such that $C^\perp \subseteq C$, we can construct an $[[n, 2k - n, d]]$ quantum code.

### 2 A variant of Construction X

Construction X is well known from the theory of classical error control codes, see for example [5, §18.7] or [1, Section 14.1]. In general this construction does not produce dual containing codes, which however is a requirement for an application of Theorem 1 to construct quantum codes. In this section we give a variant of Construction X that produces quantum codes from arbitrary quaternary linear codes.

**Theorem 2** For an $[n, k]_4$ linear code $C$ denote $e := n - k - \dim(C \cap C^\perp)$. Then there exists an $[[n + e, 2k - n + e, d]]$ quantum code with $d \geq \min\{\text{wt}(C), \text{wt}(C + C^\perp) + 1\}$.

We will now work towards the proof of Theorem 2. For $x \in \mathbb{F}_4^n$ let $||x|| = \langle x, x \rangle$ be the norm of $x$. Note that $||x|| = \sum_{i=1}^{n} x_i^2$ can only attain values 0 or 1. More specifically, $||x||$ equals the parity of $\text{wt}(x)$. Also

$$||x + y|| = ||x|| + ||y|| + \text{Tr}(\langle x, y \rangle)$$

where $\text{Tr}(a) := a + \bar{a}$ is the trace from $\mathbb{F}_4$ to $\mathbb{F}_2$. A subset $S \subseteq \mathbb{F}_4^n$ is called orthonormal if $\langle x, y \rangle = 0$ for any two distinct $x, y \in S$ and $\langle x, x \rangle = 1$ for any $x \in S$.

**Proposition 1** Let $D$ be a subspace of $\mathbb{F}_4^n$ and assume that $M$ is a basis for $D \cap D^\perp$. Then there exists an orthonormal set $B$ such that $M \cup B$ is a basis for $D$. 

Proof First note that if $S$ is a subspace of $\mathbb{F}_4^n$ such that there exist $x, y \in S$ with $(x, y) \neq 0$, then there exists $z \in S$ with $\|z\| = 1$. Indeed for $c \in \mathbb{F}_4$ we have $\|cx + y\| = \|x\| + \|y\| + \text{Tr}(c \langle x, y \rangle)$ and if $c$ assumes all values in $\mathbb{F}_4$, then $\text{Tr}(c \langle x, y \rangle)$ attains both values in $\mathbb{F}_2$. Hence there exists $c \in \mathbb{F}_4$ such that $\|cx + y\| = 1$.

Now let $W$ be a subspace of $\mathbb{F}_4^n$ such that

$$D = (D \cap D^⊥) \oplus W.$$  \hspace{1cm} (1)

Let $\ell := \text{dim}(W)$. For each $0 \leq i \leq \ell$ we will construct an orthonormal set $S_i$ that is a basis for an $i$-dimensional subspace $T_i$ of $W$ such that

$$W = T_i \oplus (T_{i+1} \cap W).$$  \hspace{1cm} (2)

The process is iterative. Take $S_0 := \emptyset$ and suppose that for some $0 \leq i < \ell$ the set $S_i$ is an orthonormal basis for $T_i$ such that $\text{dim}(T_i) = i$ and (2) holds. Let $x$ be a non-zero vector in $T_{i+1} \cap W$. Then there exists $y \in T_{i+1} \cap W$ such that $(x, y) \neq 0$. Indeed if such $y$ did not exist, then by (2) and (1) we would have that $x \in D^⊥$, which however would be a contradiction to (1) being a direct sum since we know that $x \in W$. By the first paragraph of this proof there exists $z \in T_{\ell+1} \cap W$ with $\|z\| = 1$. Set $S_{i+1} := S_i \cup \{z\}$. Clearly $S_{i+1}$ is an orthonormal set. Let $T_{i+1}$ be the subspace spanned by $S_{i+1}$. By (2) we have $z \notin T_i$, hence $\text{dim}(T_{i+1}) = i + 1$. We want to show that

$$W = T_{i+1} \oplus (T_{i+1}^⊥ \cap W).$$  \hspace{1cm} (3)

First we show that $T_{i+1} \cap T_{i+1}^⊥ \cap W = \{0\}$. Let $v \in T_{i+1} \cap T_{i+1}^⊥ \cap W$. Since $v \in T_{i+1}$, we have $v = u + cz$ where $u \in T_i$ and $c \in \mathbb{F}_4$. Since $v \in T_{i+1}^⊥$, we have for each $w \in T_i$ and each $d \in \mathbb{F}_4$

$$0 = \langle u + cz, w + dz \rangle = \langle u, w \rangle + \overline{\alpha} \langle u, z \rangle + c \langle z, w \rangle + c\overline{\alpha} \|z\| = \langle u, w \rangle + c\overline{\alpha}.$$

We must have $c = 0$ or else $\langle u, w \rangle + c\overline{\alpha}$ would not stay constant as $d$ runs through $\mathbb{F}_4$. Thus $\langle u, w \rangle = 0$ for all $w \in T_i$, hence $u \in T_{i+1}^⊥$. Since $u \in T_i$ and $T_i \cap T_{i+1}^⊥ = \{0\}$ by (2) it follows that $u = 0$. Hence $v = 0$ and $T_{i+1} \cap T_{i+1}^⊥ \cap W = \{0\}$.

Next we show that $W = T_{i+1} + (T_{i+1}^⊥ \cap W)$. Let $w \in W$. By assumption $W = T_i + (T_{i+1}^⊥ \cap W)$. Hence there exist vectors $x \in T_i$ and $y \in T_{i+1}^⊥ \cap W$ such that $w = x + y$. Let $x' := x + \langle y, z \rangle z$ and $y' := y - \langle y, z \rangle z$. Clearly $x' \in T_{i+1}$ and for any $u + dz \in T_{i+1}$ (where $u \in T_i$ and $d \in \mathbb{F}_4$) we have

$$\langle y - \langle y, z \rangle z, u + dz \rangle = \langle y, u \rangle + \overline{\alpha} \langle y, z \rangle - \langle y, \overline{\alpha} \langle y, z \rangle \rangle \langle z, u \rangle - \overline{\alpha} \|y, z\| \|z\|$$

$$= \overline{\alpha} \langle y, z \rangle - \overline{\alpha} \langle y, z \rangle = 0,$$

thus $y' \in T_{i+1}^⊥ \cap W$. Hence $W = T_{i+1} + (T_{i+1}^⊥ \cap W)$. This completes the proof that (2) implies (3) assuming that the vector $z$ is chosen as described above. Such choice of $z$ is always possible and moreover it can be done efficiently as it involves basic linear algebra. After completing the $\ell$-th step of this process, we have $T_\ell = W$ and $S_\ell$ is an orthonormal basis for $T_\ell$. This completes the proof of Proposition 1. \hfill $\square$
2.1 Proof of Theorem 2

Assume the notation as in Theorem 2. We will now prove this theorem.

Note
\[ e = \dim(C^{⊥h}) - \dim(C \cap C^{⊥h}) = \dim(C + C^{⊥h}) - \dim(C). \]

Denote \( s := \dim(C \cap C^{⊥h}) \). Let \( G \) be a block matrix
\[ G = \begin{pmatrix} M_{s \times n} & 0_{s \times e} \\ A_{(n-e-2s) \times n} & 0_{(n-e-2s) \times e} \\ B_{e \times n} & I_{e \times e} \end{pmatrix} \]
where the subscripts denote the sizes of the blocks, and \( 0, I \) are the zero matrix and the identity matrix respectively. For a matrix \( P \) let \( r(P) \) denote the set of rows of \( P \). The matrix \( G \) is constructed such that \( r(M) \) is a basis for \( C \cap C^{⊥h} \), \( r(M) \cup r(A) \) is a basis for \( C \), \( r(M) \cup r(B) \) is a basis for \( C^{⊥h} \), and \( r(B) \) is an orthonormal set. The existence of such \( B \) follows from Proposition 1. Note that \( r(M) \cup r(A) \cup r(B) \) is a basis for \( C + C^{⊥h} \).

Let \( E \) be the linear code for which \( G \) is a generator matrix. Let \( S \) denote the union of the first \( s \) rows of \( G \) and the last \( e \) rows of \( G \), i.e., \( S \) is the set of rows of the matrix
\[ \begin{pmatrix} M_{s \times n} & 0_{s \times e} \\ B_{e \times n} & I_{e \times e} \end{pmatrix}. \]
By construction, each vector in \( S \) is orthogonal to each row of \( G \), thus each vector in \( S \) belongs to \( E^{⊥h} \). Also, the vectors in \( S \) are linearly independent. Since
\[ \dim(E^{⊥h}) = n + e - (n - s) = e + s = |S| \]
it follows that \( S \) is a basis for \( E^{⊥h} \). Since \( S \) is a subset of \( E \) by construction, it follows that \( E^{⊥h} \subseteq E \).

Let \( x \) be a non-zero vector in \( E \) and in accordance with the vertical block structure of \( G \) write \( x = (x^1 \mid x^2) \) where \( x^1 \in F_4^s \) and \( x^2 \in F_4^e \). So \( x \) is a linear combination of rows of \( G \). If none of the last \( e \) rows of \( G \) enters this linear combination with a non-zero coefficient, then \( x^1 \in C \setminus \{0\} \), thus \( \text{wt}(x) = \text{wt}(x^1) \geq \text{wt}(C) \). If some of the last \( e \) rows of \( G \) enter this linear combination with a non-zero coefficient, then \( x^1 \in C + C^{⊥h} \setminus \{0\} \) and \( \text{wt}(x) = \text{wt}(x^1) + \text{wt}(x^2) \geq \text{wt}(C + C^{⊥h}) + 1 \). Thus \( E \) is an \([n + e, k + e, d]_4 \) code with \( d \geq \min\{\text{wt}(C), \text{wt}(C + C^{⊥h}) + 1\} \) and \( E^{⊥h} \subseteq E \). An application of Theorem 1 to the code \( E \) now finishes the proof of Theorem 2.

3 Finding ingredient codes

In Theorem 2 we have \( e = 0 \) if and only if \( C^{⊥h} \subseteq C \), in which case we simply obtain Theorem 1. On the other hand, if \( e \) is large then \( \text{wt}(C + C^{⊥h}) + 1 \) will be a pessimistic lower bound on the minimum weight of \( E \). Thus it seems reasonable to focus on codes for which \( e \) is positive but small. Thus characterizing such codes is an interesting problem as it leads to applications of our construction. This problem appears to be similar to one of central problems in quantum coding theory, namely characterizing dual containing (or, equivalently, self-orthogonal) linear codes, and
one may hope that similar techniques will be applicable to tackling both problems for a given class of codes. This is indeed the case for the class of cyclic codes, as we will show in this section.

For the sake of brevity we will abbreviate “linear quaternary cyclic code” to “cyclic code.” Recall that if $C$ is a cyclic code of block length $n$ with $n$ odd, then $C$ is given by a principal ideal $\langle g(x) \rangle$ in $\mathbb{F}_4[x]/(x^n - 1)$ where $g$ is called the generator polynomial for $C$. Let $m$ be the order of 4 modulo $n$, then $\mathbb{F}_{4^m}$ is the splitting field of $x^n - 1$ containing a primitive $n$th root of unity, let us call it $\beta$. Then a defining set of $C$ is the set \( \{ k : g(\beta^k) = 0, \ 0 \leq k < n \} \); note that this depends on the choice of $\beta$. We can make a canonical choice for $\beta$ by fixing a primitive element $\alpha$ of $\mathbb{F}_{4^m}$ and letting $\beta := \alpha^{(4^n - 1)/n}$. Thus we now need a canonical choice for $\alpha$; in Section 4 we let $\alpha$ be defined by the PrimitiveElement function in Magma.

For $n$ and $m$ as above and $\alpha \in \{0, \ldots, n - 1\}$ the set $\{a4^j \mod n : 0 \leq j < m \}$ is called a cyclotomic coset modulo $n$. It is well known that a defining set of a cyclic code of length $n$ is the union of some cyclotomic cosets modulo $n$. Let $\mathbb{Z}_n$ denote the set of integers modulo $n$. Clearly defining sets should be considered as subsets of $\mathbb{Z}_n$. For $S \subset \mathbb{Z}_n$ denote $\overline{S} := \mathbb{Z}_n \setminus S$ and $-2S := \{(-2s) \mod n : s \in S\}$.

**Proposition 2** If $C$ is a quaternary linear cyclic code with defining set $Z$, then $\dim(C^{+k}) - \dim(C \cap C^{+k}) = |Z \cap -2Z|$.

**Proof** Let $n$ be the block length of $C$. Let $\prod_{k \in Z}(x - \beta^k)$ be the generator polynomial for $C$, then the generator polynomial for $C^{+k}$ is $\prod_{k \in -2Z}(x - \beta^k)$ and the generator polynomial for $C \cap C^{+k}$ is $\prod_{k \in \mathbb{Z} \cup -2Z}(x - \beta^k)$. Then

$$
\dim(C^{+k}) - \dim(C \cap C^{+k}) = n - |\mathbb{Z} - 2\mathbb{Z}| - (n - |\mathbb{Z} \cup -2\mathbb{Z}|) \\
= |\mathbb{Z} \cup -2\mathbb{Z}| - |\mathbb{Z} - 2\mathbb{Z}| = |Z \cap -2Z|.
$$

$\square$

The fact that any defining set is the union of some cyclotomic cosets modulo $n$ allows us to design a simple backtracking algorithm to enumerate all cyclic codes with a given block length and a given upper bound on the value $\dim(C^{+k}) - \dim(C \cap C^{+k}) = |Z \cap -2Z|$ which is denoted $e$ in Theorem 2. We start with $Z = \emptyset$ and we add one cyclotomic coset to $Z$ at a time. We note that if $Z' \supseteq Z$, then $|Z' \cap -2Z| \geq |Z \cap -2Z|$, which gives an easy backtracking rule for enforcing an upper bound on $|Z \cap -2Z|$.

### 3.1 A special lower bound for certain cyclic codes

In cases when the defining set of a cyclic code has a certain special structure we can prove the following refinement of Theorem 2. Note that if $n$ is divisible by 3, then $\{0\}, \{\frac{n}{3}\}$ and $\{\frac{2n}{3}\}$ are singleton cyclotomic cosets modulo $n$ and moreover each of them is fixed under the map $S \mapsto -2S$ that operates on subsets $S$ of $\mathbb{Z}_n$.

**Theorem 3** Assume that $n$ is divisible by 3 and let $C$ be an $[n,k]_4$ cyclic code with defining set $Z$ such that $Z \cap -2Z \subseteq \{0, \frac{n}{3}, \frac{2n}{3}\}$. Denote $e := |Z \cap -2Z|$. Then there exists an $[[n + e, 2k - n + e, d]]$ quantum code with $d \geq \min\{\wt(C), \wt(C_u) + 1, \wt(C + C^{+k}) + 2\}$ where the minimum is taken over the cyclic codes $C_u$ with defining set $Z \setminus \{u\}$ for each $u \in Z \cap -2Z$. 

Proof Denote \( n = 3\ell \) and let \( \omega \) denote a primitive cube root of unity in \( \mathbb{F}_4 \). For \( t \in \{0,1,2\} \) define the polynomials

\[
b_t(x) := \frac{x^{3\ell} - 1}{x - \beta^{\frac{4t}{3}\ell}} = \frac{x^{3\ell} - 1}{x - \omega^t} = \sum_{i=0}^{\ell-1} \left(x^{3i+2} + \omega^t x^{3i+1} + \omega^{2t} x^{3i}\right).
\]

As is common in the theory of cyclic codes, with a polynomial \( \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_4[x]/(x^n - 1) \) we associate the vector \( (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_4^n \) and with a slight abuse of notation we use the same symbol for both objects. Note that \( \{b_0, b_1, b_2\} \) is an orthonormal set since \( n \) is odd.

We now follow the structure of the proof of Theorem 2, where for the rows of \( B \) we take the set \( U = \{b_t : \frac{4t}{3}\ell \in Z \cap -2Z\} \). Note that \( b_t \in C^\perp \setminus C \) for each \( b_t \in U \) and the vectors in \( U \) are linearly independent, as required in the first part of the proof of Theorem 2.

For the second part of the proof (the claim about the minimum distance of the quantum code), let \( E \) be the quaternary linear code as introduced in the proof of Theorem 2. Let \( x \) be a non-zero vector in \( E \) and consider \( x \) as a linear combination of rows of \( G \). This time we split the proof into three cases: (i) No rows of \( B \) enter the linear combination with a non-zero coefficient, (ii) exactly one row of \( B \) enters the linear combination with a non-zero coefficient, (iii) at least two rows of \( B \) enter the linear combination with a non-zero coefficient. In cases (i) and (iii) the argument is the same as in the proof of Theorem 2, with the obvious adjustment of the term +1 to +2. In case (ii) let \( b_t \) denote the row of \( B \) that enters the linear combination with a non-zero coefficient. Then \( x \) belongs to the span of \( C \) and \( b_t \), which is easily seen to be precisely the cyclic code with the defining set \( Z \setminus \{\frac{4t}{3}\ell\} \); note that \( b_t \) is divisible by \( \prod_{k \in Z \setminus \{\frac{4t}{3}\ell\}} (x - \beta^k) \).

\( \square \)

4 New quantum codes

Just as a case study, in order to demonstrate that our results do produce quantum codes with better parameters than hitherto known codes, we applied the backtracking algorithm outlined in the previous section to enumerate cyclic codes \( C \) with \( e \leq 3 \) and modest block length. Then all codes appearing in the lower bounds of Theorems 2 and 3 are cyclic. We computed the minimum weight of these codes using the built-in function in Magma \cite{Magma}. In Table 1 we report some examples of new quantum codes whose minimum distance is higher than the minimum distance of the best known quantum code with the same length and dimension as recorded in \cite{Hong}. We denote by \( Cy(n; a_1, \ldots, a_r) \) the cyclic code with block length \( n \) and the defining set \( \bigcup_{i=1}^{r} C_{a_i} \), where \( C_a \) denotes the cyclotomic coset modulo \( n \) containing \( a \).

It should be noted that secondary constructions such as \cite[Theorem 6]{Hong} applied to codes listed in Table 1 produce many more record breaking codes; for example our \([53, 17, 10]\) code alone leads to at least 28 new record breaking codes improving the lower bounds of \cite{Hong}.

In the course of our computations we have also come across dual containing cyclic codes that produce quantum codes improving the lower bounds of \cite{Hong} merely by applying Theorem 1; these codes are listed in Table 2.
Construction X for quantum error-correcting codes

Table 1

<table>
<thead>
<tr>
<th>quantum code</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>[52, 26, 7]</td>
<td>Cy(51; 0, 1, 6, 35)</td>
</tr>
<tr>
<td>[53, 17, 10]</td>
<td>Cy(51; 0, 1, 2, 6, 17, 22)</td>
</tr>
<tr>
<td>[54, 8, 12]</td>
<td>Cy(51; 0, 1, 3, 9, 17, 22, 34, 35)</td>
</tr>
<tr>
<td>[54, 24, 8]</td>
<td>Cy(51; 0, 1, 3, 17, 34, 35)</td>
</tr>
<tr>
<td>[65, 31, 9]</td>
<td>Cy(63; 0, 2, 3, 11, 15, 31, 42)</td>
</tr>
<tr>
<td>[86, 56, 8]</td>
<td>Cy(85; 0, 7, 30, 34, 57)</td>
</tr>
<tr>
<td>[92, 56, 8]</td>
<td>Cy(91; 0, 7, 30, 34, 57)</td>
</tr>
<tr>
<td>[92, 30, 14]</td>
<td>Cy(91; 0, 2, 3, 9, 14, 34)</td>
</tr>
<tr>
<td>[92, 48, 10]</td>
<td>Cy(91; 0, 1, 14, 19, 39)</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>quantum code</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>[86, 9, 19]</td>
<td>Cy(85; 3, 7, 9, 10, 17, 19, 21, 30, 37, 57)</td>
</tr>
<tr>
<td>[85, 13, 17]</td>
<td>Cy(85; 3, 10, 13, 19, 21, 29, 30, 37, 57)</td>
</tr>
<tr>
<td>[93, 3, 21]</td>
<td>Cy(93; 1, 5, 9, 13, 17, 23, 33, 34, 45)</td>
</tr>
</tbody>
</table>

5 Conclusion

We have presented a construction of stabilizer quantum codes from linear quaternary codes that do not need to satisfy the dual-containment (equivalently, self-orthogonal) requirement. We have given a general lower bound on the distance of the resulting quantum code, as well as a more specialized lower bound applicable to certain cyclic codes. We have shown that these lower bounds produce record breaking codes. We noted that it is an interesting problem to characterize classes of codes that are “nearly missing” the dual-containment condition, as such codes can be used as ingredients to our construction. We provided such characterization for cyclic codes.

Also, one should note that there is a considerable freedom in choosing the orthonormal set \( B \) whose existence is proved in Proposition 1, and in general different choices of set \( B \) will lead to inequivalent codes. A randomized method for constructing the set \( B \) can be extracted from our proof of Proposition 1; this can be then used in searches for good codes.

References

Gregor Leander
Ruhr Universität, Buchum, Germany

Design and Analysis of Block Cipher - Links to Boolean Functions and Coding Theory

Abstract:
In this talk I will start with motivating why there is still a need for the design of new block ciphers. After that I am going to present the recently proposed block cipher PRINCE. In the last part I am going to highlight some open issues that arise when designing a new block cipher and discuss relations to coding theory and Boolean functions.
Low Rank Parity Check codes and their application to cryptography

Philippe Gaborit · Gaétan Murat · Olivier Ruatta · Gilles Zémor

Abstract. In this paper we introduce a new family of rank metric codes: the Low Rank Parity Check codes for which we propose an efficient probabilistic decoding algorithm. This family of codes can be seen as the equivalent of classical LDPC codes for the rank metric. We then propose to use these codes for cryptography in the McEliece encryption setting. At the difference of previous encryption algorithms based on rank metric -especially based on Gabidulin codes-, the codes we use have a very poor structure. Our cryptosystem can be seen as an equivalent to the NTRU cryptosystem [16] (and also to the more recent MDPC[23] cryptosystem) in a rank metric context. Overall our system permits to achieve a very low public key of 1517 bits for a security of $2^{80}$, moreover our system is very fast, with a decryption failure which can be made arbitrarily small.

Keywords: Public key cryptosystem, rank metric

1 Introduction

The rank metric was introduced by Gabidulin in 1985 in [8], along with the Gabidulin codes which are an equivalent of the Reed-Solomon codes for the rank metric. Since then, rank metric codes have been used in many applications: for coding theory and space-time codes and also for cryptography. Until now the main tool for rank based cryptography was based on masking the Gabidulin codes [10] in different ways and using the McEliece (or Niederreiter) setting with these codes. Meanwhile most of the systems were broken by using structural attacks which used the particular structure of the Gabidulin codes ([27], [6], [2], [19],[9]). A similar situation exists in the Hamming case for which all cryptosystems based on the Reed-Solomon have been broken for the same reason: the Reed-Solomon are so structured that their structure is difficult to mask and there is always structural information leaking.

P. Gaborit, G. Murat and O. Ruatta
Université de Limoges, XLIM-DMI,
123, Av. Albert Thomas
87000 Limoges Cedex, France.
E-mail: gaborit.murat.ruatta@unlim.fr

G. Zémor
Université Bordeaux I, A2X,
351 cours de la Liberté
33400 Talence Cedex, France.
E-mail: gilles.zemor@math.u-bordeaux.fr
Since the introduction of code-based cryptography by McEliece in 1978, the different cryptosystems, proposed in the Hamming distance setting, were based on masking a special family of decodable codes, like Goppa, Reed-Muller of Reed-Solomon codes. The strong structure of these codes usually implies a large size of public key. Now in 1996 and 1997, two lattice-based cryptosystems were proposed independently: the NTRU [16] and the GGH [15] cryptosystems which can be seen as a cryptosystems in a McEliece setting but for the Euclidean distance. Notice that lattice based cryptography is no more than code-based cryptography with q-ary codes but with the Euclidean distance rather than the Hamming distance. Both NTRU and GGH cryptosystems are based on the same idea: knowing a random basis of small weight vectors permits to obtain an efficient decoding algorithm suitable for cryptography. Moreover the NTRU cryptosystem (which can be seen as an optimized case of the GGH cryptosystem [21]) introduced for the first time the idea to use double-circulant matrices in order to decrease the size of the public key, this idea was made possible because of the randomness of the small dual basis. At last we remark that for 15 years the NTRU cryptosystem has not been really attacked on its double-circulant structure, indeed the best attacks still remain general LLL attacks on lattice.

In a classical cryptographic Hamming context, the first author [11] introduced in 2005 the idea to use quasi-cyclic codes to decrease the size of the public, meanwhile the idea to add a quasi-cyclic structure on an already structured family of codes introduces too much structure and the system was broken [25]. This idea was then used with other families of quasi-cyclic (or quasi-dyadic) structured codes like Goppa quasi-dyadic [22] or quasi-cyclic alternant codes [1], these systems lead to much smaller keys, but eventually some parameters were attacked in [5] and even though the idea remains valid, the cryptanalysis of [5] showed that this the idea of quasi-cyclic or quasi-dyadic structured codes could not lead to secure public key of a few thousand bits, but rather to secure keys of a few ten thousand bits.

More recently new proposal were made in the spirit of the original NTRU schemes with Hamming distance, first by the use of quasi-cyclic LDPC codes, then with MDPC codes in [23]. The last family of codes permits to obtain the same type of feature than the NTRU cryptosystem: a very compact key (of 4800b) and a security based on decoding by a random dual matrix with small weight.

Our contribution In this paper we built anew on the NTRU setting but in a rank metric context. We introduce the Low Rank Parity Check codes (an equivalent of the LDPC codes for Hamming distance) for which we propose an efficient probabilistic decoding algorithm. We then use these codes in a quasi-cyclic form and obtain a cryptosystem with public key three times smaller than the MDPC codes (1517 bits).

The paper is organized as follows: Section 2 gives background on rank metric codes and cryptography, Section 3 consider results on subspaces, Section 4 defines the LRPC codes, Section 5 gives a decoding algorithm and at last Section 6 and 7 consider the cryptographic application of these codes.

2 Background on rank metric codes and cryptography

2.1 Definitions and notation

Notation:
Let \( q \) be a power of a prime \( p \), \( m \) an integer and let \( V_n \) be a \( n \) dimensional vector space over the finite field \( \text{GF}(q^m) = F_{q^m} \). Let \( \beta = (\beta_1, \ldots, \beta_m) \) be a basis of \( F_{q^m} \) over \( F_q \). Let \( F_i \) be the map from \( F_{q^m} \) to \( F_q \) where \( F_i(x) \) is the \( i \)-th coordinate of \( x \).
in the basis $\beta$. To any $v = (v_1, \ldots, v_n)$ in $V_n$ we associate the matrix $\overline{v} \in M_{m,n}(F_q)$ in which $\overline{v}_{i,j} = F_1(v_j)$. The rank weight of a vector $v$ can be defined as the rank of the associated matrix $\overline{v}$. If we name this value $\text{rank}(v)$ we can have a distance between two vectors $x, y$ using the formula $\text{rd}(x, y) = \text{rank}(x - y)$. We refer to [20] for more details on codes for the rank distance.

A rank code $C$ of length $n$ and dimension $k$ over $F_{q^m}$ is a subspace of dimension $k$ of $F_{q^m}$ embedded with its rank metric. The minimum rank distance of the code $C$ is the minimum rank of non-zero vectors of the code.

**Definition 1** Let $x = (x_1, x_2, \cdots, x_n) \in F_{q^m}^n$ be a vector of rank $r$. We denote by $E$ the $F_q$-sub vector space of $F_{q^m}$ generated by $x_1, x_2, \cdots, x_n$. The vector space $E$ is called the **support** of $x$.

**Remark 1** The notion of support of a code word for Hamming distance and the one introduced in definition 1 are different even if they share a common principle. Indeed, in both case, giving a low weight syndrome associated to $x$, once the support is known one only have to solve a linear system in both case.

**Remark 2** For any code, the action of the general linear group does not change the weight of the words. In the case a rank metric code, this action does not change the supports of the word also. Also an interesting remark is that in the case of Hamming distance over $F_q$ increasing the value of $q$ permits to increase the minimum distance of a code but does not change the type of support (it is always a binomial coefficient with the same length) when for rank metric increasing the base field $F_q$ increases in a strong the size of the support (ie: the number of bases which can be found by the Gaussian binomial).

**Notation 1** In the following, $C$ is a rank metric code of length $n$ and dimension $k$ over $F_{q^m}$. The matrix $G$ denotes a $k \times n$ generator matrix of $C$ and $H$ one of its parity check matrix.

### 2.2 Cryptography and rank codes

The main problem used for rank codes in the cryptographic context is the generalization of the classical syndrome decoding problem with Hamming distance in the case of rank metric:

**Rank syndrome decoding problem (RSD)** Let $H$ be a $( (n-k) \times n)$ matrix over $F_{q^m}$ with $k \leq n$, $s \in F_{q^m}^k$ and $r$ an integer. The problem is to find $x$ such that $\text{rank}(x) = r$ and $Hx^t = s$.

In that case it is not proven that the problem is $NP$-hard, but this problem is very close to the syndrome decoding problem which is $NP$-hard, moreover the problem can be seen as a structured version of the MinRank problem which is also $NP$-hard (the RSD problem can be attacked as a MinRank problem but in practice the attack does not work since there are too many unknowns [4]). Moreover the problem has been studied for more than 20 years and the best attacks are exponential, so that the problem is generally believed to be hard.

The first non-trivial attack on the problem was proposed by Chabaud and Stern [3] in 1996, then in 2002 Ourivski and Johannson [26] improved the previous attack and proposed a new attack, meanwhile these two attacks did not take account of the value of $m$ in the exponent. Very recently the two previous attacks were generalized
The product space \( \langle AB \rangle \) is obviously generated by the set 
\[ \{ A_1, B_1, \ldots, A_{\alpha}, B_{\beta} \} \]
and its dimension is bounded above by \( \alpha \beta \).

A question of interest in our case is the probability that the dimension is not maximal when \( \alpha \) and \( \beta \) are relatively small. We suppose \( \alpha \beta < m \) and we investigate the typical dimension of the product subspace \( \langle AB \rangle \).

Let \( A \) and \( B \) be random \( F_q \)-subspaces of \( F_q^m \) of dimensions \( \alpha \) and \( \beta \) respectively. We suppose \( \alpha \beta < m \) and we investigate the typical dimension of the subspace \( \langle AB \rangle \).

We rely on the following observation:

**Lemma 1** Let \( A' \) and \( B \) be two subspaces of \( F_q^m \) of dimensions \( \alpha' \) and \( \beta \) such that \( \dim(A') = \alpha' \beta \). Let \( A = A' + \langle a \rangle \) where \( a \) is a uniformly chosen random element of \( F_q^m \). Then

\[
\mathbb{P}(\dim(AB) < \alpha' \beta + \beta) \leq \frac{q^{\alpha' \beta + \beta}}{q^m}.
\]

**Proposition 1** Let \( B \) be a fixed subspace and suppose we construct a random subspace \( A \) by choosing \( \alpha \) independent (in the sense of probability) random vectors of \( F_q^m \) and letting \( A \) be the subspace generated by these \( \alpha \) random vectors. We have that \( \dim(AB) = \alpha \beta \) with probability at least \( 1 - \alpha q^{\alpha \beta} \).

Let \( B \) be a fixed subspace of \( F_q^m \) containing 1 and let \( \langle B^2 \rangle \) be the subspace generated by all products of elements of \( B \). Let \( \beta_2 = \dim(B^2) \). Let \( A \) be a random subspace of \( F_q^m \) of dimension \( \alpha \). By Proposition 1 we have that \( \dim(AB^2) = \alpha \beta_2 \) with probability at least \( 1 - \alpha q^{\alpha \beta_2} \). Remark that we have \( \beta_2 \leq \beta(\beta + 1)/2 \).

**Lemma 2** Suppose \( \dim(AB^2) = \alpha \beta_2 \). Let \( e \in \langle AB \rangle \) with \( e \notin A \). Suppose \( eB \subset \langle AB \rangle \). Then there exists \( x \in B \) such that \( xB \subset B \).
**Proposition 2** Suppose $m$ is prime. Let $A$ and $B$ be random subspaces of dimensions $\alpha$ and $\beta$ respectively. Let $(b_i)$ be a basis of $B$ and let $S = \langle AB \rangle$. Then with probability at least
$$1 - \alpha q^{\frac{\alpha(\beta + 1)}{2}}$$ we have that $\bigcap_i b_i^{-1} S = A$.

**Proposition 3** Let $B$ be a subspace of dimension $\beta$ containing 1 such that $\dim B + Bb^{-1} = 2\beta - 1$ for some $b \in B$. Let $A$ be a randomly chosen subspace of dimension $\alpha$. With probability at least
$$1 - \alpha \frac{q^{\beta(2\beta - 1)}}{q^{\beta}}$$ we have that $\langle AB \rangle \cap \langle AB \rangle b^{-1} = A$.

**Remark 3** It is interesting to remark that in practice the probability for which an upper bound is given in Proposition 2 and 3, decreases much more faster to 0. Indeed when the degree of the extension $m$ increases by one (for $m$ greater than $rd$), the probability that $\bigcap_i b_i^{-1} S \neq A$ is divided by a factor at least $q^{r-1}$. This means that in practice the previous upper bound is rather bad, and that one can consider that as soon as $m$ is greater than $rd$ by 8 or more (and increasing) the probability is far below $2^{-30}$. It will be the case when one will choose parameters in the last section.

### 4 Low Rank Parity Check Codes

**4.1 Definition of Low Rank Parity Check codes**

The idea of these codes is to generalize the classical LDPC codes approach for Hamming distance to the rank metric. There is a natural analogy between low density matrices and matrices with low rank.

**Definition 3** A Low Rank Parity Check (LRPC) code of rank $d$, length $n$ and dimension $k$ over $F_q$ is a code such that the code has for parity check matrix, a $(n - k) \times n$ matrix $H(h_{ij})$ such that the sub-vector space of $F_q$ generated by its coefficients $h_{ij}$ has dimension at most $d$. We call this dimension the weight of $H$. Denoting $F$ the sub-vector space of $F_q$ generated by the coefficients $h_{ij}$ of $H$, we denote by $\{F_1, F_2, \cdots, F_d\}$ one of its basis.

In practice it means that for any $1 \leq i \leq n - k; 1 \leq j \leq n$, there exist $h_{ij} \in F_q$ such that $h_{ij} = \sum_{l=1}^{d} h_{ij} F_l$. One can also define a special sub-class of LRPC codes:

**Definition 4** A Quasi-Cyclic Low Rank Parity Check (QC-LRPC) code of rank $d$, is a quasi-cyclic code such that the code has for parity check matrix $H$, a quasi-cyclic matrix of low weight $d$.

**Remark 4** Of particular interest is the case of double circulant LRPC codes (DC-LRPC) of rank $d$, which are codes with parity check matrix $H$, a double-circulant matrix (concatenation of two cyclic matrices) of weight $d$.

**4.2 Writing the syndrome equations in the base field for LRPC codes**

The particular structure of LRPC codes permits to express formally the syndrome equations in $F_q^m$ in equations over the base field $F_q$, it permits to obtain a very efficient decoding algorithm, that we will see in the next section. We describe in the following the way to obtain such a transformation, in particular we introduce a matrix $A_H$, that will be used for the decoding procedure.
Suppose that the error \( e(e_1, \ldots, e_n) \) of weight \( r \) lie in the error space \( E \) of dimension \( r \) generated by a basis \( \{E_1, \ldots, E_r\} \), then all \( e_i \) (1 \( \leq \) \( i \) \( \leq \) \( n \)) can be written as 
\[ e_i = \sum_{j=1}^{d} e_{ij} E_j. \]

The matrix \( H = (h_{ij}) \) is constructed such that \( h_{ij} \) belong to a space \( F \) of dimension \( d \) generated by \( \{F_1, \ldots, F_d\} \), then for all \( 1 \leq i \leq n-k, 1 \leq j \leq n \), 
\[ h_{ij} = \sum_{l=1}^{d} h_{ijl} F_l, \]
for \( h_{ij} \in F_q \). Suppose moreover that the dimension of the space \( <F_1E_1, F_1E_2, \ldots, F_1Er, F_2E_1, \ldots, F_dEr> \) is exactly \( rd \), it is then possible to express the syndrome equations \( H.e^t = s \) over \( F_{qm} \) into equations over \( F_q \), formally expressing the \( e_i \) in the basis \( \{E_1, \ldots, E_r\} \) and the syndrome coordinates in the product basis \( \{F_1E_1, F_1E_2, \ldots, F_1Er, F_2E_1, \ldots, F_dEr\} \).

More formally there exists a \((n-k)rd \times nr\) matrix \( A_H^r \) over \( F_q \) such that the syndromes equations \( H.e^t = s \) over \( F_{qm} \) become a system of equations \( A_H^r.e^t = s' \) in \( F_q \), where \( s' \) corresponds to a \((n-k)rd\) vector in \( F_q \) corresponding to the syndrome coordinates of \( s \), \((s_1, \ldots, s_{n-k})\), where each \( s_i \) is written in the basis \( \{F_1E_1, F_1E_2, \ldots, F_1Er, F_2E_1, \ldots, F_dEr\} \) of the product space \( <E,F> \), and where \( e^t \) is a \( nr \) vector over \( F_q \) corresponding to the error coordinates \( (e_1, \ldots, e_n) \), where each \( e_i \) is written in the basis \( \{E_1, \ldots, E_r\} \). If one proceeds this way the \( nr \times (n-k)rd \) matrix \( A_H^r \) only depends on the coordinates \( h_{ij} \) written in the basis \( \{F_1, \ldots, F_d\} \) and therefore \( A_H^r \) does not depend on the error \( e \), uniquely of its dimension \( r \). For different errors \( e \), one just has to write \( s \) as \( s' \) in the product space \( <E,F> \) and then solve the system \( A_H^r.e^t = s' \). The previous construction can be seen as formally unfolding the matrix \( H \) in the basis of the product space: \( \{F_1E_1, F_1E_2, \ldots, F_1Er, F_2E_1, \ldots, F_dEr\} \).

**Definition 5 (Description of the matrix \( A_H^r \))** With previous notation consider a \((n-k)rd \times nr\) matrix \( A_H^r = (a_{ij}) \). We first set all \( a_{ij} = 0 \) and then write:

\[ a_{u+(v-1)r+(i-1)rd,u+(j-1)r} = h_{ijv}, \]

for \( 1 \leq u \leq r, 1 \leq i \leq n-k, 1 \leq j \leq n \) and \( 1 \leq v \leq d \).

As explained before, the matrix \( A_H^r \) corresponds to a symbolic rewriting of the system \( H.e^t = s \) in the base field \( F_q \) in the basis \( \{F_1E_1, F_1E_2, \ldots, F_1Er, F_2E_1, \ldots, F_dEr\} \) and \( \{E_1, \ldots, E_r\} \).

For instance, the first row of \( A_H^r \) consists of the impact of the error vector \( e(e_1, \ldots, e_n) \) on the first row of \( H \), \((h_{11}, \ldots, h_{1n})\) regarding the symbolic basis element \( F_1E_1 \).

Now if one takes random values of coordinates for low rank \( H \), it is easy to find matrices \( A_H^r \) of maximal rank \( nr \).

**Definition 6** In the following we denote by \( A_H \) a \( nr \times nr \) invertible submatrix of \( A_H^r \), and we denote by \( D_H = A_H^{-1} \) a decoding matrix of \( H \).

**Remark 5** The matrix \( D_H \) permits to recover directly the \( nr \) values \( e_{ij} \) from \( nr \) positions of the \( s_i \) written in product basis by a simple multiplication.

### 5 Decoding algorithm for LRPC codes

#### 5.1 General idea

The general idea of the algorithm is to use the fact that the weight of the parity check matrix is small, the idea is that the space generated by the coordinates of
syndrome $< s_1, \ldots, s_{n_k} >$ permits to recover the whole product space $P = \langle E, F \rangle$ of the support of the error and of the known small basis of $H$. Knowing the whole space $P$ and the space $F$ permits to recover $E$. Then, knowing the support $E$ of the error $e$, it is easy to recover the exact value of each coordinate by solving a linear system. This approach is very similar to the classical decoding procedure of BCH codes for instance, where one recovers the error locator polynomial, which gives the support of the error, and then the value of the error coordinates.

5.2 A general decoding algorithm

Consider a $[n, k]$ LRPC code $C$ of low weight $d$ over $F_q^m$, with generator matrix $G$ and dual $(n - k) \times n$ matrix $H$, such that all the coordinates $h_{ij}$ of $H$ belong to a space $F$ of rank $d$ with basis $\{ F_1, \ldots, F_d \}$ and suppose that as in the previous section $H$ is chosen such that there exists an invertible associated decoding matrix $D_H$.

Suppose the received word to be $y = xG + e$ for $x$ and $e$ in $(F_q^m)^n$, and where $e(e_1, \ldots, e_n)$ is the error vector of rank $r$, which means that for any $1 \leq i \leq n$, $e_i \in E$, a vector space of dimension $r$ with basis (say) $\{ E_1, \ldots, E_r \}$.

We have the following general decoding algorithm, this algorithm has a probability of failure that we will consider in the next subsection, we give general parameters at the end of the section for which the algorithm works.

![Algorithm 1](image)

Fig. 1 Algorithm 1: a general decoding algorithm for LRPC codes

5.3 Correctness of the algorithm

We prove the correctness of the algorithm in the ideal case when dimension $(\langle E, F \rangle) = rd$, dimension $(S) = rd$ and dimension $(S_1 \cap S_2 \cap \cdots \cap S_d) = r$, we will see in the next subsection that this is the general case.

**step 1:** The first step of the algorithm is obvious.

**step 2:** now we want to prove that $E \subseteq S_1 \cap S_2 \cap \cdots \cap S_d$. We defined $S_i = F_i^{-1}S = \{ F_i^{-1}x, x \in S \}$, now since by hypothesis $S$ is exactly the product space $E.F = \{ a.b | a \in E, b \in F \}$, we have $F_i.E_j \in S, \forall 1 \leq j \leq r$, hence $E_j \in S_i$, and
therefore \( E \subset S_i \), and hence \( E \) is contained in \( S_1 \cap S_2 \cap \cdots \cap S_d \); now by hypothesis \( \dim(S_1 \cap S_2 \cap \cdots \cap S_d) = \dim(E) \) and hence \( E = S_1 \cap S_2 \cap \cdots \cap S_d \).

**Step 3:** once the support \( E \) of the error of \( x \) is known, one can write \( x = \sum_{i \leq s, 1 \leq j \leq r} e_{ij} E_j \), for \( e_{ij} \in F_r \) and solve the linear system \( H \cdot x^t = s \) in the \( nr \) unknowns \( e_{ij} \). The system has \( nr \) unknowns in \( F_q \) and \((n-k)m\) equations in \( F_q \) coming from the \( n-k \) syndrome equations in \( F_q^m \). The parameter \( r \) is chosen such that \( r \geq \frac{(n-k)m}{q} \). Notice moreover that one can consider the product space \( <E,F> \) for a formal \( F \) so that in that case the system equations are uniquely related to the matrix \( H \). Hence \( H \) can be chosen such that a decoding matrix \( D_H \), defined in the previous section exists, it considerably decreases the complexity of the decoding since the decoding is obtained by a simple matrix multiplication rather than by a linear system solving.

5.4 Probability of failure

We now consider the different possibility of failure, there three cases to consider. The case \( \dim(<E,F>) = rd \) corresponds to Prop. 1 of Section 3, the case \( E = S_1 \cap S_2 \cap \cdots \cap S_d \) corresponds to Prop. 2 of the same section. In both cases the probability can be made exponentially small depending on parameters, especially when in practice the upper bound given are really large compared to experimental results.

The last case is the case \( \dim(S) = rd \). We have the following easy proposition:

**Proposition 4** The probability that the \( n-k \) syndromes does not generate the product space \( P = <E,F> \) is less than \( q^{1+(n-k)-rd} \).

**Proof** By construction all \( s_i \) belong to the product space \( P \) and since the error is taken randomly the \( s_i \) can be seen as random elements of \( P \), now if one considers a set of \((n-k)\) random elements of space of dimension \( rd \) (with \( n-k \geq rd \)) the probability that this set does not generate the whole space is roughly given by \( q^{-(1+(n-k)-rd)} \) the probability that a random \([rd,n-k] = [rd,rd+(n-k)-rd] \) matrix over \( F_q \) not be invertible.

Therefore the previous discussion shows that depending on the parameters the probability of failure of the previous algorithm can be made arbitrarily small and that the main probability we have to consider in fact is the probability given by Proposition 4, which is not an upper bound but what happens in practice.

5.5 Complexity of decoding

The most costly step of the algorithm are step 2) et step 3). The cost of step 2) is the cost of the intersection of vector spaces which has cost \( 4r^2d^2nm \) operations in the base field (this operation can also be done in a very elegant way with \( q \)-polynomials [24]). Now the cost of step 3) is the cost of solving the system \( H \cdot e^t = s \) when the support \( E \) of the error is known, if one proceeds naively there are \( nr \) unknowns (the \( e_{ij} \)) and the cost of matrix inversion in \( n^3r^3 \), now one can use the formal decoding matrix \( D_H \) of the previous section and simply recover the \((e_{ij})\) by multiplying by \( D_H \) the \( nr \) positions (written in the product basis of \( (E,F) \)) of \( s_1,..,s_{n-k} \) associated to the matrix \( D_H \) of definition 6. Therefore the cost of the inversion becomes only the cost.
of a matrix multiplication: \( n^2r^2 \). Remark that the matrix \( D_H \) can be precomputed and stocked or even reconstructed column by column from random hash values - in that case one fixes \( D_H \) and one derives \( H \).

5.6 A general theorem

If we sum up the results of the different subsections one obtains the following general theorem:

**Theorem 1** Let \( H \) be a \((n-k) \times n\) dual matrix of a LRPC codes with low rank \( d \geq 2 \) over \( F_{q^m} \), then algorithm 1 decodes a random error \( e \) of low rank \( r \) such that \( rd \leq n-k \), with failure probability \( q^{-(n-k+1-rd)} \) and complexity \( r^2(4d^2m+n^2) \).

**Proof** This theorem is a direct result from previous subsection.

**Remark 6** In term of pure decoding capacity the LRPC codes are less interesting than Gabidulin codes, since they hardly decode up to \((n-k)/2\) and moreover the algorithm is probabilistic, but they are perfectly fitted for cryptography.

6 Application to cryptography: the LRPC cryptosystem

In the following we propose a new cryptosystem in the spirit of NTRU and the more recent MDPC system.

6.1 The LRPC cryptosystem

For our new cryptosystem we use the McEliece cryptographic setting, the Niederreiter setting could also be used but less semantic security is known for this setting.

Let us consider \( C \) a LRPC code with a \((n-k) \times n\) parity check matrix \( H \). We consider \( H \) to be either a LRPC codes or double circulant LRPC codes (DC-LRPC) \( \frac{n}{2} \times \frac{n}{2} \) of rank \( d \), such that the code corrects error of rank \( r \). We hide the matrix \( H \) with a random invertible \( \frac{n}{2} \times \frac{n}{2} \) matrix \( R \), in the case of double circulant codes the matrix \( S \) is random circulant \( \frac{n}{2} \times \frac{n}{2} \). Figure 2 presents the LRPC cryptosystem.

1. **Key creation** Choose a random LRPC code over \( F_{q^m} \) of low rank \( d \) with support \( F \) and parity check \((n-k) \times n\) matrix \( H \), generator matrix \( G \) and decoding matrix \( D_H \) which correct errors of rank \( r \) (as in previous section). And a random invertible \((n-k) \times (n-k)\) matrix \( R \)
   - **Secret key**: the low rank matrix \( H \), the masking matrix \( R \)
   - **Public key**: the matrix \( G' = RG \)
2. **Encryption** Translate the information vector \( M \) into a word \( x \), choose a random error \( e \) of rank \( r \) on \( F_{q^m} \). The encryption of \( M \) is \( c = xG' + e \).
3. **Decryption** Compute syndrome \( s = H.c \) and recover the error vector \( e \) by decoding the LRPC code, then \( xG' = c - e \) and \( x \).

![Fig. 2](image-url) The LRPC cryptosystem
Remark 7 The cryptosystem can be adapted in the case of DC-LRPC codes, in that case the matrix $G'$ can be written $G' = (A^{-1}B^T I)$ were $A$ and $B$ are two circulant matrices of low rank $d$ for the same space $F$.

- General parameters of the LRPC cryptosystem:
Writing the system in systematic form we obtain:

1. Size of public key (bits): LRPC: $(n - k)k \cdot \log(q)$ / DC-LRPC: $\frac{nm}{2} \cdot \log(q)$
2. Size of secret key (bits): a random vector can used to recover the different parameters
3. Size of message: LRPC: $nm \cdot \log(q)$ / DC-LRPC: $nm \cdot \log(q)$
4. Encryption rate: LRPC: $\frac{k}{n}$ / DC-LRPC: $\frac{1}{2}$
5. Complexity of encryption: LRPC: $k(n - k)$ multiplications in $F_q^m$ / DC-LRPC: $\frac{n^2}{2}$ multiplications in $F_q^m$.
6. Complexity of decryption: $k^2$ multiplications in $F_q^m$.
7. Complexity of the best usual attack: Support attack: $O((n - k)^2 m^3 q^{(r - 1)}(\frac{(k + 1)m}{n}))$ / algebraic attacks (Gröbner basis) - lower bound: $q^r(\frac{(k + 1) - (n + 1)}{r})$ and heuristic results of [18]

Remark 8 For decryption and encryption, the most costly operations are the matrix-vector multiplication and the cost of syndrome computing (in particular in the cryptosystem setting, the syndrome computation is more expensive than the cost of decoding LRPC - especially because of the hiding matrix $R$). In the case of DC-LRPC, one can use the double-circulant structure to improve computations. The cost of a multiplication in the extension field $F_q^m$ is $m \cdot \log_2(m) \cdot \log_2(\log_2(m))$ ([14]). In general for the LRPC cryptosystem we take $r = d$ which gives the previous best known attacks.

Remark 9 We choose to present a McEliece setting, in that case the size of the message is greater than for the Niederreiter setting but more can be proven for semantic security.

We made a non-optimized implementation in Magma which confirmed our results.

7 Security of the LRPC cryptosystem

7.1 Semantic security

The problem on which relies the security of our system is the following:

**The LRPC problem:** Given a public matrix $G'$ it is difficult to recover low weight vector of rank weight $d$ in the dual code.

**Discussion on the problem** The problem considered here is the adaptation of the NTRU problem and the MDPC problem but in rank metric. Notice that clearly the matrix used are not random, meanwhile this special structure could not be used for attacks for NTRU over 15 years. Also for MDPC the same situation arises.

Now in term of semantic security the approach developed for the MDPC cryptosystem in [23] on the indistinguishability to random codes can be adapted in a context of rank metric, moreover particular the CCA-2 conversion of K. Kobara and H. Imai
[17], can also be adapted to rank metric but this discussion goes well beyond this extended abstract.

Also concerning decryption failure it is possible to use the approach of E. Fujisaki and T. Okamoto [7] which permits that no information is given in case of decryption failure (the same approach was proposed for NTRU and MDPC).

7.2 Practical security

For parameters \( r \) and \( d \) we can choose \( d = r \) (although they may be different). It means in that case that an attacker will attack two different codes but with small vectors with the same low weight \( r = d \). We review the different known attacks:

- **Message attack**: in that case the attacker tries to recover directly the message \( M \) by trying to recover \( e \) of rank \( r \) with classical attacks on a random code.

- **Structural attacks on the secret key**: the attacker tries to recover the structure of the hidden code, by searching a codeword of rank \( d \) in a parity check matrix \( H \) of \( G' \). Notice that classical attacks first recover the support of a small weight word, in that case all the rows of \( H \) have the same support and the fact that there are \( n/2 \) cyclic vector do not seem to be helpful - in particular for rank metric adding different rows with the same support does not change the support.

- **Spurious key attack**: as in the NTRU case (see [16]) this attack corresponds to finding small word vectors in \( H \) with rank slightly greater than \( d \), and to use them for decoding. Although theoretically possible this attack is not doable in practice since the fact that \( H \) contains small weight vectors implies indeed that many words of weight \( 2d \) exist. We do not go into details in this extended abstract but as for MDPC codes [23], when the weight increases the complexity of the attacks grows faster than the number of small weight vectors, so that this attacks - as for NTRU and MDPC- does not work in practice.

Overall no structural attacks seem to appear in that case. Notice that this system is the exact adaptation of the NTRU and GGH frame in the case of rank distance. In particular, the double circulant case with \( G' = (A^{-1}B|I) \) corresponds to this case and no attack was found.

8 Examples of parameters

We give three examples of parameters for the DC-LPRC case: an example with security \( 2^{80} \) operations which optimizes the size of the public key at 1500\( b \) with a decryption probability of \( 2^{-22} \), an example with security \( 2^{128} \), and at last an example with decryption failure probability of \( 2^{-80} \). In the table ‘failure’ stands for probability of ‘decryption failure’, the size of the public key is in bits, the security is in bits. We give parameters for different level of security, but also for different decryption failure, in particular it is possible to reach a \( 2^{-80} \) easily at the cost of doubling the size of the key. Notice that the parameters are very versatile. Although no special attack is known for non prime number we choose to consider prime numbers in general. The complexity of decryption for the first set of parameters is \( 2^{20} \) bit operations. In particular in terms of computation cost the LRPC cryptosystem seems to compare very well with the MDPC cryptosystem.
9 Conclusion

In this paper, as the recent MDPC paper [23] we generalize the NTRU [16] approach in a coding theory context but with the rank metric. To do so we introduce a new type of codes, the LRPC codes, for which we propose an efficient decoding algorithm. Overall as it is often the case for rank metric codes, the new cryptosystem has very interesting features, like having very small keys and being fast and efficient, and compares well to Hamming distance cryptosystems. Moreover when rank metric cryptosystems have a strong history of being broken because of their structure (indeed all previously known rank-based cryptosystems are based on hiding the Gabidulin code structure), the cryptosystem we propose is the first rank-metric based cryptosystem with a poor random structure and which is not based on Gabidulin codes. It is interesting to remark that this type of poor structure in the double-circulant case was never really attacked in the case of lattices as it seems the case for the MDPC cryptosystem. Of course the cryptosystem needs more scrutiny from the community but it seems to be a very interesting cryptosystem for the future.

References

10. Ernst M. Gabidulin, A. V. Paramonov, O. V. Tretjakov: Ideals over a Non-Commutative Ring and other Applications in Cryptology. EUROCRYPT 1991: 482-489

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>m</th>
<th>q</th>
<th>d</th>
<th>r</th>
<th>failure</th>
<th>public key</th>
<th>security</th>
</tr>
</thead>
<tbody>
<tr>
<td>74</td>
<td>37</td>
<td>41</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>-22</td>
<td>1517</td>
<td>80</td>
</tr>
<tr>
<td>94</td>
<td>47</td>
<td>47</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>-23</td>
<td>2307</td>
<td>128</td>
</tr>
<tr>
<td>68</td>
<td>34</td>
<td>23</td>
<td>2^9</td>
<td>4</td>
<td>4</td>
<td>-80</td>
<td>3128</td>
<td>100</td>
</tr>
</tbody>
</table>

18. F. Levy-dit-Vehel and L. Perret, Algebraic decoding of rank metric codes, proceedings of YACCC06.


Distinguisher-Based Attacks on Public-Key Cryptosystems Using Reed-Solomon Codes

Alain Couvreur\(^1\), Philippe Gaborit\(^2\), Valérie Gauthier\(^3\), Ayoub Otmani\(^4\), and Jean-Pierre Tillich\(^5\)

\(^1\) GRACE Project, INRIA Saclay & LIX, CNRS UMR 7161 - École Polytechnique, 91120 Palaiseau Cedex, France. alain.couvreur@lix.polytechnique.fr
\(^2\) XLIM, CNRS UMR 7252 - Université de Limoges, 123 avenue Albert Thomas, 87060 Limoges Cedex, France. philippe.gaborit@unilim.fr
\(^3\) Departamento de Matemáticas, Universidad de los Andes, Bogotá Colombia ve.gauthier@uniandes.edu.co
\(^4\) Normandie Univ, France; UR, LITIS, F-76821 Mont-Saint-Aignan, France. ayoub.otmani@univ-rouen.fr
\(^5\) SECRET Project - INRIA Rocquencourt, 78153 Le Chesnay Cedex, France. jean-pierre.tillich@inria.fr

Abstract. The purpose of this paper is to demonstrate that a distinguisher of Reed-Solomon codes based on the square code construction leads to the cryptanalysis of several cryptosystems relying on them. These schemes are respectively (i) a homomorphic encryption scheme proposed by Bogdanov and Lee; (ii) a variation of the McEliece cryptosystem proposed by Baldi et al. which firstly uses Reed-Solomon codes instead of Goppa codes and secondly, adds a rank 1 matrix to the permutation matrix; (iii) Wieschebrink’s variant of the McEliece cryptosystem which consists in concatenating a few random columns to a generator matrix of a secretly chosen generalized Reed-Solomon code.

1 Introduction

The first cryptographic scheme using generalized Reed-Solomon codes was proposed in 1986 by Niederreiter [Nie86] but it was shown to be insecure in [SS92]. The attack recovers the underlying Reed-Solomon code allowing the decoding of any encrypted data. However during the past years there were several attempts to repair this scheme. The first one was proposed by Wieschebrink [Wie06] and consists in choosing a generator matrix of a generalized Reed-Solomon code and adding to it a few random columns. It was advocated that this modification avoids the Sidelnikov-Shestakov attack [SS92]. The second one is another variant of McEliece’s cryptosystem [McE78] proposed in [BBC+11] which uses this time a generator matrix of a generalized Reed-Solomon but hides its structure differently than in the McEliece cryptosystem: instead of multiplying by a permutation matrix, the generator matrix is multiplied by a matrix whose inverse is of the form \( \Pi + R \) where \( \Pi \) is a permutation matrix and \( R \) is a rank 1 matrix. The key point of this modification is that the public code obtained with this method is not anymore a generalized Reed-Solomon code and this seems to thwart the Sidelnikov and Shestakov attack completely. More recently, some of the nice algebraic properties of the Reed-Solomon codes were also used to devise the first public-key homomorphic encryption scheme [BL11] based on coding theory.

Contrarily to the Niederreiter’s proposal [Nie86] based on generalized Reed-Solomon codes, the original McEliece cryptosystem [McE78] which uses Goppa codes, has withstood many key-recovery attacks and after more than thirty years now, it still belongs to the very few unbroken public-key cryptosystems. No significant breakthrough has been observed with respect to the problem of recovering the private key. For instance, the weak keys found in [Gib91,LS01] can be easily avoided. This fact has led to claim that the generator matrix of a binary Goppa code does not disclose any visible structure that an attacker could exploit. This is strengthened by the fact that Goppa codes share many characteristics with random codes.
However, in [FGO+11], an algorithm that manages to distinguish between a random code and a Goppa code has been introduced. This work, without undermining the security of [McE78], prompts to wonder whether it would be possible to devise an attack based on such a distinguisher. It turns out [MCP12] that the distinguisher in [FGO+11] has an equivalent but simpler description in terms of the component-wise product of codes. This notion was first put forward in coding theory to unify many different algebraic decoding algorithms [Pel92,Köt92]. Recently, it was used in [MCMMP11a,MCMMP12a] to study the security of cryptosystems based on Algebraic-Geometric codes. Powers of codes are also studied in the context of secure multi-party computation (see for example [CCCX09,CCX11]). This distinguisher is even more powerful in the case of Reed-Solomon codes than for Goppa codes because, whereas for Goppa codes it is only successful for rates close to 1, it can distinguish Reed-Solomon codes of any rate from random codes.

In the specific case of [BL11], the underlying public code is a modified Reed-Solomon code obtained from the insertion of a zero submatrix in the Vandermonde generating matrix defining it and in this case, the aforementioned distinguisher leads to an attack. More exactly, we present a key-recovery attack on the Bogdanov-Lee homomorphic scheme based on the version of our distinguisher presented in [MCP12]. Our attack runs in polynomial time and is efficient: it only amounts to calculate the ranks of certain matrices derived from the public key. In [BL11] the columns that define the zero submatrix are kept secret and form a set $L$. We give here a distinguisher that detects if one or several columns belong to $L$ or not. It is constructed by considering the code generated by component-wise products of codewords of the public code (the so-called “square code”). This operation is applied to punctured versions of this square code obtained by picking a subset $I$ of the whole set indexing the columns. It turns out that the dimension of the punctured square code is directly related to the cardinality of the intersection of $I$ with $L$. This gives a way to recover the full set $L$ allowing the decryption of any ciphertext.

To demonstrate further the power of this approach, we propose another cryptanalysis against the variant of McEliece’s cryptosystem [McE78] proposed in [BBC+11]. As explained above, the public code obtained with this method is not anymore a generalized Reed-Solomon code. On the other hand, it contains a very large secret generalized Reed-Solomon code. We present an attack that is based on a distinguisher which is able to identify elements of this secret code. This distinguisher is again derived from considerations about the dimension of component-wise products of codes. Once this secret code is obtained, it is then possible to completely recover the initial generalized Reed-Solomon code by using the square-code construction as in [Wie10]. We are then able to decode any ciphertext.

Finally, we also cryptanalyze the first variant of the McEliece’s cryptosystem based on Reed-Solomon codes [Wie06]. We show here how a refinement of our distinguisher permits to recover the random columns added to the generator matrix of the generalized Reed-Solomon code. Once these column positions are recovered, the Sidelnikov and Shestakov attack can be used on the non-random part of the generator matrix to completely break the scheme.

It should also be pointed out that the properties of Reed-Solomon codes with respect to the component-wise product of codes have already been used to cryptanalyze a McEliece-like scheme [BL05] based on subcodes of Reed-Solomon codes [Wie10]. The use of this product is nevertheless different in [Wie10] from the way we use it here. Note also that our attack is not an adaptation of the Sidelnikov and Shestakov approach [SS92]. Our approach is completely new: it illustrates how a distinguisher that detects an abnormal behavior can be used to recover a private key.
2 Reed-Solomon Codes and the Square Code

We recall in this section a few relevant results and definitions from coding theory and bring in the fundamental notion which is used in both attacks, namely the square code. Generalized Reed-Solomon codes (GRS in short) form a special case of codes with a very powerful low complexity decoding algorithm. It will be convenient to use the definition of these codes as evaluation codes.

**Definition 1 (Reed-Solomon code and generalized Reed-Solomon code).** Let \( k \) and \( n \) be integers such that \( 1 \leq k < n \leq q \) where \( q \) is a power of a prime number. The generalized Reed-Solomon code \( \text{GRS}_k(x,y) \) of dimension \( k \) is associated to a tuple \((x,y) \in \mathbb{F}_q^n \times \mathbb{F}_q^n\) where \( x \) is an \( n \)-tuple of distinct elements of \( \mathbb{F}_q \) and the entries \( y_i \) are arbitrary non-zero elements in \( \mathbb{F}_q \). It is defined as \( \text{GRS}_k(x,y) \defeq \{(y_1p(x_1),\ldots,y_n p(x_n)) : p \in \mathbb{F}_q[X], \deg p < k \}. \) Reed-Solomon codes correspond to the case where \( y_i = 1, \forall i. \)

It has been suggested to use them in a public-key cryptosystem for the first time in [Nie86] but it was discovered that this scheme is insecure in [SS92]. Sidelnikov and Shestakov namely showed that it is possible to recover in polynomial time for any GRS code a tuple \((x,y)\) which is denoted by \( a \). Solomon codes (GRS in short) form a special case of codes with a very powerful low complexity decoding algorithm. It will be convenient to use the definition of these codes as evaluation codes.

**Definition 2 (Star product of codes – Square code).** Let \( \mathcal{A} \) and \( \mathcal{B} \) be two codes of length \( n \). The star product code denoted by \( \mathcal{A} * \mathcal{B} \) of \( \mathcal{A} \) and \( \mathcal{B} \) is the vector space spanned by all products \( a*b \) where \( a \) and \( b \) range over \( \mathcal{A} \) and \( \mathcal{B} \) respectively. When \( \mathcal{B} = \mathcal{A}, \mathcal{A} * \mathcal{A} \) is called the square code of \( \mathcal{A} \) and is denoted by \( \mathcal{A}^2 \).

It is clear that \( \mathcal{A} * \mathcal{B} \) is also generated by the \( a_i \ast b_j \)'s where the \( a_i \)'s and the \( b_j \)'s form a basis of \( \mathcal{A} \) and \( \mathcal{B} \) respectively. Therefore

**Proposition 1.**

\[
\dim(\mathcal{A} * \mathcal{B}) \leq \dim(\mathcal{A}) \dim(\mathcal{B}).
\]

We expect that the square code when applied to a random linear code should be a code of dimension of order \( \min \left\{ \binom{k+1}{2}, n \right\} \). Actually it can be shown by the proof technique of [FGO+11] that with probability going to 1 as \( k \) tends to infinity, the square code is of dimension \( \min \left\{ \binom{k+1}{2} (1 + o(1)), n \right\} \) when \( k \) is of the form \( k = o(n^{1/2}) \). On the other hand, GRS codes behave in a completely different way.

**Proposition 2.** \( \text{GRS}_k(x,y)^2 = \text{GRS}_{2k-1}(x,y \ast y). \)

This follows immediately from the definition of a GRS code as an evaluation code since the star product of two elements \( c = (y_1p(x_1),\ldots,y_np(x_n)) \) and \( c' = (y_1q(x_1),\ldots,y_n q(x_n)) \) of \( \text{GRS}_k(x,y) \) where \( p \) and \( q \) are two polynomials of degree at most \( k - 1 \) is of the form

\[
c \ast c' = (y_1^2p(x_1)q(x_2),\ldots,y_n^2p(x_n)q(x_n)) = (y_1^2r(x_1),\ldots,y_n^2 r(x_n))
\]

where \( r \) is a polynomial of degree at most \( 2k - 2 \). Conversely, any element of the form \( (y_1^2r(x_1),\ldots,y_n^2 r(x_n)) \) where \( r \) is a polynomial of degree less than or equal to \( 2k - 2 \) is a linear combination of star products of two elements of \( \text{GRS}_k(x,y). \)
This proposition shows that when $2k - 1 \leq n$, the square code is only of dimension $2k - 1$, which is abnormally small. This property can also be used in the case $2k - 1 > n$. To see this, consider the dual of the Reed-Solomon code itself a Reed-Solomon code [MS86, Theorem 4, p.304]

**Proposition 3.**

$$\text{GRS}_k(x, y) = \text{GRS}_{n-k}(x, y')$$

where the length of $\text{GRS}_k(x, y)$ is $n$ and $y'$ is a certain element of $\mathbb{F}_q^n$ depending on $x$ and $y$.

Therefore when $2k - 1 > n$ a Reed-Solomon code $\text{GRS}_k(x, y)$ can also be distinguished from a random linear code of the same dimension by computing the dimension of $\langle (\text{GRS}_k(x, y))^\perp \rangle^2$. We have in this case $\langle (\text{GRS}_k(x, y))^\perp \rangle^2 = \langle \text{GRS}_{n-k}(x, y') \rangle^2 = \langle \text{GRS}_{2n-2k-1}(x, y' \star y') \rangle$ and we obtain a code of dimension $2n - 2k - 1$.

The star product of codes has been used for the first time by Wieschebrink to cryptanalyze a McEliece-like scheme [BL05] based on subcodes of Reed-Solomon codes [Wie10]. The use of the star product is nevertheless different in [Wie10] from the way we use it here. In Wieschebrink’s paper, the star product is used to identify for a certain subcode \(C\) of the square of various punctured versions of the public code in the Baldi et al. case.

- the square of various punctured versions of the public code in the Bogdanov and Lee case in order to retrieve the columns which correspond to the Reed-Solomon part;
- or identifying a certain subcode of the public code by this means in the Baldi et al. case.

### 3 The Bogdanov-Lee Homomorphic Cryptosystem

#### 3.1 The scheme

The cryptosystem proposed by Bogdanov and Lee in [BL11] is a public-key homomorphic encryption scheme based on linear codes. It encrypts a plaintext \(m\) from \(\mathbb{F}_q\) into a ciphertext \(c\) that belongs to \(\mathbb{F}_q^n\) where \(n\) is a given integer. The key generation requires a non-negative integer \(\ell\) such that \(3\ell < n\) and a subset \(L\) of \(\{1, \ldots, n\}\) of cardinality \(3\ell\). A set of \(n\) distinct elements \(x_1, \ldots, x_n\) from \(\mathbb{F}_q\) are generated at random. They serve to construct a \(k \times n\) matrix \(G\) whose \(i\)-th column \(G_{i}^T\) is defined by:

\[
G_{i}^T = \begin{cases} 
(x_i, x_i^2, \ldots, x_i^{\ell}, 0, \ldots, 0) & \text{if } i \in L \\
(x_i, x_i^2, \ldots, x_i^{\ell}, x_i^{\ell+1}, \ldots, x_i^n) & \text{if } i \notin L 
\end{cases}
\]

where the symbol \(T\) stands for the transpose. The cryptosystem is defined as follows.

1. **Secret key.** \((L, G)\).
2. **Public key.** \(P \overset{\text{def}}{=} SG\) where \(S\) is a \(k \times k\) random invertible matrix over \(\mathbb{F}_q\).
3. **Encryption.** The ciphertext \(c \in \mathbb{F}_q^n\) corresponding to \(m \in \mathbb{F}_q\) is obtained as \(c \overset{\text{def}}{=} xP + m1 + e\) where \(1 \in \mathbb{F}_q^n\) is the all-ones row vector, \(x\) is picked uniformly at random in \(\mathbb{F}_q^k\) and \(e\) in \(\mathbb{F}_q^n\) by choosing its components according to a certain distribution \(\eta\).
4. **Decryption.** The linear system (1) is solved for $y \defeq (y_1, \ldots, y_n) \in \mathbb{F}_q^n$:

$$G y^T = 0, \sum_{i \in L} y_i = 1 \text{ and } y_i = 0 \text{ for all } i \notin L. \tag{1}$$

Then the plaintext is $m = \sum_{i=1}^{n} y_i c_i$.

The decryption algorithm will output the correct plaintext when $\ell$ and $n$ are chosen such that the entry $e_i$ at position $i$ of the error vector is zero when $i \in L$. The distribution $\eta$ which is used to draw at random the coordinates of $e$ is chosen such that this property holds with very large probability. More precisely, the parameters $k$, $q$, $\ell$ and the noise distribution $\tilde{\eta}$ are chosen such that $q = \Omega \left( 2^{n^{\alpha}} \right)$, $k = \Theta \left( n^{1-\alpha/8} \right)$, $\ell = \Theta \left( n^{\alpha/4} \right)$ and the noise distribution $\tilde{\eta}$ is the $q$-ary symmetric channel with noise rate $\eta = \Theta \left( 1/n^{1-\alpha/4} \right)$ where $\alpha$ is a in $\left( 0, \frac{1}{4} \right]$ (for more details see [BL11, §2.3]. It is readily checked that the probability that $e_i \neq 0$ for $i \in L$ is vanishing as $n$ goes to infinity since it is upper-bounded by $\eta \ell = \Theta \left( \frac{n^{\alpha/4}}{n^{1-\alpha/4}} \right) = \Theta \left( n^{-1+\alpha/2} \right) = o(1)$.

### 3.2 An Efficient Attack on the Bogdanov-Lee Scheme

The attack consists in first recovering the secret set $L$ and from here we find directly a suitable vector $y$ by solving the system

$$P y^T = 0, \sum_{i \in L} y_i = 1, y_i = 0 \text{ for all } i \notin L. \tag{2}$$

Indeed, requiring that $P y^T = 0$ is equivalent to $S G y^T = 0$ and since $S$ is invertible this is equivalent to the equation $G y^T = 0$. Therefore System (2) is equivalent to the “secret” system (1). An attacker may therefore recover $m$ without even knowing $G$ just by outputting $\sum_{i} y_i c_i$ for any solution $y$ of (2). In the following subsection, we will explain how $L$ can be recovered from $P$ in polynomial time.

**Recovering $L$.** Our attack which recovers $L$ relies heavily on the fact that the public matrix may be viewed as the generator matrix of a code $C$ which is quite close to a generalized Reed-Solomon code (or to a Reed-Solomon code if a row consisting only of 1’s is added to it). Notice that any punctured version of the code has also this property (a punctured code consists in keeping only a fixed subset of positions in a codeword). More precisely, let us introduce

**Definition 3.** For any $I \subset \{1, \ldots, n\}$ of cardinality $|I|$, the restriction of a code $\mathcal{A}$ of length $n$ is the subset of $\mathbb{F}_q^{|I|}$ defined as $\mathcal{A}_I \defeq \{ v \in \mathbb{F}_q^{|I|} | \exists a \in \mathcal{A}, v = (a_i)_{i \in I} \}$.

The results about the unusual dimension of the square of a Reed-Solomon codes which are given in Section 2 prompt us to study the dimension of the square code $< C^2_\ell >$ or more generally the dimension of $< C^2_I >$. When $I$ contains no positions in $L$, then $C_I$ is nothing but a generalized Reed-Solomon code and we expect a dimension of $2k-1$ when $|I|$ is larger than $2k-1$. On the other hand, when there are positions in $I$ which also belong to $L$ we expect the dimension to become bigger and the dimension of $< C^2_I >$ to behave as an increasing function of $|I \cap L|$. This is exactly what happens as shown in the proposition below.

---

$^6$ It means that $\text{Prob}(e_i = 0) = 1 - \eta$ and $\text{Prob}(e_i = x) = \frac{n}{q-1}$ for any $x$ in $\mathbb{F}_q$ different from zero.
Proposition 4. Let $I$ be a subset of $\{1, \ldots, n\}$ and set $J \stackrel{\text{def}}{=} I \cap L$. If the cardinality of $I$ and $J$ satisfy $|J| \leq \ell - 1$ and $|I| - |J| \geq 2k$ then

$$\dim(<\varepsilon_J^2>) = 2k - 1 + |J|.$$  \hspace{1cm} (3)

The proof of this proposition can be found in [GOT12a, Appendix A]. An attacker can exploit this proposition to mount a distinguisher that recognizes whether a given position belongs to the secret set $L$. At first a set $I$ which satisfies with high probability the assumptions of Proposition 4 is randomly chosen. Take for instance $|I| = 3k$. Then $d_I \stackrel{\text{def}}{=} \dim(<\varepsilon_I^2>)$ is computed. Next, one element $x$ is removed from $I$ to get a new set $I'$ and $d_{I'} = \dim(<\varepsilon_{I'}^2>)$ is computed. The only two possible cases are either $x \notin L$ then $d_{I'} = d_I$ or $x \in L$ and then $d_{I'} = d_I - 1$. By repeating this procedure, the whole set $J = I \cap L$ is easily recovered. The next step now is to find all the elements of $L$ that are not in $I$. One solution is to exchange one element in $I \setminus J$ by another element in $\{1, \ldots, n\} \setminus I$ and compare the values of $d_I$. If it increases, it means that the new element belongs to $L$. At the end of this procedure the set $L$ is totally recovered. This probabilistic algorithm is obviously of polynomial time complexity and breaks completely the homomorphic scheme suggested in [BL11].

4 Baldi et al. Variant of McEliece’s Cryptosystem

4.1 The scheme

The cryptosystem proposed by Baldi et al. in [BBC+11] is a variant of McEliece’s cryptosystem [McE78] which replaces the permutation matrix used to hide the secret generator matrix by one of the form $\Pi + R$ where $\Pi$ is a permutation matrix and $R$ is a rank-one matrix. From the authors’ point of view, this new kind of transformation would allow to use families of codes that were shown insecure in the original McEliece’s cryptosystem. In particular, it would become possible to use GRS codes in this new framework. The scheme can be summarized as follows.

Secret key.
- $G_{\text{sec}}$ is a generator matrix of a GRS code of length $n$ and dimension $k$ over $\mathbb{F}_q$;
- $Q \stackrel{\text{def}}{=} \Pi + R$ where $\Pi$ is an $n \times n$ permutation matrix;
- $R$ is a rank-one matrix over $\mathbb{F}_q$ such that $Q$ is invertible. In other words there exist $\alpha \stackrel{\text{def}}{=} (\alpha_1, \ldots, \alpha_n)$ and $\beta \stackrel{\text{def}}{=} (\beta_1, \ldots, \beta_n)$ in $\mathbb{F}_q^n$ such that $R = \alpha^T \beta$.
- $S$ is a $k \times k$ random invertible matrix over $\mathbb{F}_q$.

Public key. $G_{\text{pub}} \stackrel{\text{def}}{=} S^{-1}G_{\text{sec}}Q^{-1}$.

Encryption. The ciphertext $c \in \mathbb{F}_q^n$ of a plaintext $m \in \mathbb{F}_q^k$ is obtained by drawing at random $e$ in $\mathbb{F}_q^n$ of weight less than or equal to $\frac{n-k}{2}$ and computing $c \stackrel{\text{def}}{=} mG_{\text{pub}} + e$.

Decryption. It consists in performing the three following steps:
1. Guessing the value of $eR$;
2. Calculating $c' \stackrel{\text{def}}{=} cQ - eR = mS^{-1}G_{\text{sec}} + eQ - eR = mS^{-1}G_{\text{sec}} + e\Pi$ and using the decoding algorithm of the GRS code to recover $mS^{-1}$ from the knowledge of $c'$;
3. Multiplying the result of the decoding by $S$ to recover $m$.

The first step of the decryption, that is guessing the value $eR$, boils down to trying $q$ elements (in the worst case) since $eR = e\alpha^T \beta = \gamma \beta$ where $\gamma$ is an element of $\mathbb{F}_q$. 

4.2 Attack on the Baldi et al. Cryptosystem when $2k + 2 < n$

We define $C_{\text{sec}}$ and $C_{\text{pub}}$ to be the codes generated by the matrices $G_{\text{sec}}$ and $G_{\text{pub}}$ respectively. We denote by $n$ the length of these codes and by $k$ their dimension. We assume in this subsection that
\[ 2k + 2 < n \]

As explained in Subsection 4.1, $C_{\text{sec}}$ is a GRS code. It will be convenient to bring in the code $C_{\text{def}} = C_{\text{sec}} \Pi^{-1}$. From [GOT12b, Lemma 3, Appendix A], the matrix $R \Pi^{-1}$ is also of rank one. Hence there exist $a$ and $b$ in $\mathbb{F}_q^n$ such that:
\[ R \Pi^{-1} = b^T a. \]  

This code $C_{\text{def}}$, being a permutation of a GRS code, is itself a GRS code. So there are elements $x$ and $y$ in $\mathbb{F}_q^n$ such that $C_{\text{def}} = \text{GRS}_k(x, y)$. There is a simple relation between $C_{\text{pub}}$ and $C_{\text{def}}$ as explained by the following lemma.

**Lemma 1.** Let $\lambda = -\frac{1}{1 + a \cdot b}$. For any $c$ in $C_{\text{pub}}$ there exists $p$ in $C$ such that:
\[ c = p + (p \cdot \lambda)a. \]  

The proof of this lemma is given in [GOT12b, Lemma 2, §4.2]. From now on we make the assumption that
\[ \lambda \notin C_{\text{def}}^{-1}. \]  

If this is not the case then $C_{\text{pub}} = C = \text{GRS}_k(x, y)$ and there is straightforward attack by applying the Sidelnikov and Shestakov algorithm [SS92]. It finds $(x', y')$ that expresses $C_{\text{pub}}$ as $\text{GRS}_k(x', y')$. Our attack relies on identifying a code of dimension $k - 1$ that is both a subcode of $C_{\text{pub}}$ and the GRS code $C$. It consists more precisely of codewords $p + (p \cdot \lambda)a$ with $p$ in $C$ such that $p \cdot \lambda = 0$. This particular code which is denoted by $C_{\lambda^{-1}}$ is therefore:
\[ C_{\lambda^{-1}} = C \cap < \lambda >^{-1} \]

where $< \lambda >$ denotes the vector space spanned by $\lambda$. It is a subspace of $C_{\text{pub}}$ of codimension 1 if $\lambda \notin C_{\text{def}}^{-1}$. This strongly suggests that $< C_{\text{pub}}^2 >$ should have an unusual low dimension since $< C_{\text{def}}^2 >$ has dimension $2k - 1$ by Proposition 2. More exactly we have here:

**Proposition 5.**
1. $< C_{\text{pub}}^2 > \subset < C^2 > + C \ast a + < a \ast a >$
2. $\dim \left( < C_{\text{pub}}^2 > \right) \leq 3k - 1$

The first fact follows immediately from Lemma 1 and the proof of this proposition is given in [GOT12b, Appendix A]. Experimentally it has been observed that the upper-bound on the dimension is sharp. Indeed, the dimension of $< C_{\text{pub}}^2 >$ has always been found to be equal to $3k - 1$ in all our experiments when choosing randomly the codes and $Q$.

The second observation is that when a basis $g_1, \ldots, g_k$ of $C_{\text{pub}}$ is chosen and $l$ other random elements $z_1, \ldots, z_l$, then we may expect that the dimension of the vector space generated by all products $z_i \ast g_j$ with $i$ in $\{1, \ldots, l\}$ and $j$ in $\{1, \ldots, k\}$ is the dimension of the full space $< C_{\text{pub}}^2 >$ when $l \geq 3$. This is indeed the case when $l \geq 4$ but it is not true for $l = 3$ since we have the following result.
success is 1 of the attack, namely finding a suitable triple \( z \) same test to check whether the triple \( z \) repeat each test in the \( C \) GRS code that we want to recover. This is exactly the problem which was solved in [Wie10]. In getting any other element of \( C \) \( C \) in \( B \) and if \( T \) \( 12: \)

Proposition 7. \( \text{Let } z \text{ is in } C_{\lambda}^\perp \text{ for } i \in \{1, 2, 3\} \text{ then for all } j \in \{1, \ldots, k\}: \)

\[
z_i \ast g_j \subset < g^2 > + < z_1 \ast a > + < z_2 \ast a > + < z_3 \ast a >
\]

and if \( B \) is the linear code spanned by \( \{z_i \ast g_j \mid 1 \leq i \leq 3 \text{ and } 1 \leq j \leq k\} \) then

\[
\dim(B) \leq 2k + 2.
\]

The proof of this proposition is straightforward and is given in [GOT12b, Appendix A]. The upper-bound given in (9) on the dimension follows immediately from (8). This leads to Algorithm 1 which computes a basis of \( C_{\lambda}^\perp \). It is essential that the condition in (4) holds in order to distinguish the case when the dimension is less than or equal to \( 2k + 2 \) from higher dimensions. The first phase of the attack, namely finding a suitable triple \( z_1, z_2, z_3 \) runs in expected time \( O(k^3q^3) \) because each test in the repeat loop 1 has a chance of \( \frac{1}{q} \) to succeed. Indeed, \( C_{\lambda}^\perp \) is of codimension 1 in \( C_{\text{pub}} \) and therefore a fraction \( \frac{1}{q} \) of elements of \( C_{\text{pub}} \) belongs to \( C_{\lambda}^\perp \). Once \( z_1, z_2, z_3 \) are found, getting any other element of \( C_{\lambda}^\perp \) is easy. Indeed, take a random element \( z \in C_{\text{pub}} \) and use the same test to check whether the triple \( z_1, z_2, z \) is in \( C_{\lambda}^\perp \). Since \( z_1, z_2 \in C_{\lambda}^\perp \) the probability of success is \( \frac{1}{q} \) and hence \( z \) can be found in \( O(q) \) tests. The whole algorithm runs in expected time \( O(k^3q^3) + O(k^4q) = O(k^3q^3) \) since \( k = O(q) \) and the first phase of the attack is dominant in the complexity. Once \( C_{\lambda}^\perp \) is recovered, it still remains to recover the secret code and \( a \). The problem at hand can be formulated like this: we know a very large subcode, namely \( C_{\lambda}^\perp \) of a GRS code that we want to recover. This is exactly the problem which was solved in [Wie10].

Algorithm 1 Recovering \( C_{\lambda}^\perp \).

Input: A basis \( \{g_1, \ldots, g_k\} \) of \( C_{\text{pub}} \).

Output: A basis \( L \) of \( C_{\lambda}^\perp \).

1: repeat
2: for \( 1 \leq i \leq 3 \) do
3: Randomly choose \( z_i \) in \( C_{\text{pub}} \)
4: end for
5: \( B \leftarrow < \{z_i \ast g_j \mid 1 \leq i \leq 3 \text{ and } 1 \leq j \leq k\} > \)
6: until \( \dim(B) \leq 2k + 2 \) and \( \dim(< z_1, z_2, z_3 >) = 3 \)
7: \( L \leftarrow \{z_1, z_2, z_3\} \)
8: \( s \leftarrow 4 \)
9: while \( s \leq k - 1 \) do
10: repeat
11: Randomly choose \( z_s \) in \( C_{\text{pub}} \)
12: \( \mathcal{F} \leftarrow < \{z_i \ast g_j \mid i \in \{1, 2, s\} \text{ and } 1 \leq j \leq k\} > \)
13: until \( \dim(\mathcal{F}) \leq 2k + 2 \) and \( \dim(< \mathcal{L} \cup \{z_s\}>) = s \)
14: \( L \leftarrow \mathcal{L} \cup \{z_s\} \)
15: \( s \leftarrow s + 1 \)
16: end while
17: return \( L \);
our case this amounts to compute $<C^2_{\lambda} >$ which turns out to be equal to $\text{GRS}_{2k-1}(x, y * y)$ (see [MCMMP11b,MCMMP12b] for more details). It suffices to use the Sidelnikov and Shestakov algorithm [SS92] to compute a pair $(x, y * y)$ describing $<C^2_{\lambda} >$ as a GRS code. From this, we deduce a pair $(x, y)$ defining the secret code $C$ as a GRS code. The final phase, that is, recovering a possible $(\lambda, a)$ pair and using it to decode the public code $C_{\text{pub}}$, is detailed in [GOT12b, Appendix B].

4.3 Using duality when rates are larger than $\frac{1}{2}$

The codes suggested in [BBC+11, §5.1.1,§5.1.2] are all of rate significantly larger than $\frac{1}{2}$, for instance Example 1 p.15 suggests a GRS code of length 255, dimension 195 over $\mathbb{F}_{256}$, whereas Example 2. p.15 suggests a GRS code of length 511, dimension 395 over $\mathbb{F}_{512}$. The attack suggested in the previous subsection only applies to rates smaller than $\frac{1}{2}$. There is a simple way to adapt the previous attack for this case by considering the dual $C_{\text{pub}}^\perp$ of the public code. Note that by Proposition 3, there exists $y'$ in $\mathbb{F}_n$ for which we have $C_{\text{pub}}^\perp = \text{GRS}_{n-k}(x, y')$. Moreover, $C_{\text{pub}}^\perp$ displays a similar structure as $C_{\text{pub}}$.

Lemma 2. For any $c$ from $C_{\text{pub}}^\perp$ there exists an element $p$ in $C^\perp$ such that:

$$c = p + (p \cdot a)b. \quad (10)$$

The proof of this lemma is given in [GOT12b, Appendix A]. It implies that the whole approach of the previous subsection can be carried out over $C_{\text{pub}}^\perp$. It allows to recover the secret code $C_{\text{pub}}^\perp$ and therefore also $C$. This attack needs that $2(n-k) + 2 < n$, that is $2k > n + 2$. In summary, there is an attack as soon as $k$ is outside a narrow interval around $n/2$ which is $[\frac{n-2}{2}, \frac{n+2}{2}]$. We have implemented this attack on magma for $n = 127$, $q = 2^7$, $k = 30$ and the average running time over 50 attacks was about 9 hours.

5 Wieschebrink’s Scheme

In [Wie06] Wieschebrink suggests a variant of the McEliece cryptosystem based on GRS codes whose purpose was to resist to the Sidelnikov–Shestakov attack. The idea of this proposal is to use the generator matrix of a GRS code in which a small number of randomly chosen columns are inserted. More precisely, let $G$ be a generator matrix of a GRS code of length $n$ and dimension $k$ defined over $\mathbb{F}_q$. Let $C_1, \ldots, C_r$ be $r$ column vectors in $\mathbb{F}_q^k$ drawn uniformly at random and let $G'$ be the matrix obtained by concatenating $G$ and the columns $C_1, \ldots, C_r$. Choose $S$ to be a $k \times k$ random invertible matrix and let $Q$ be a an $(n+r) \times (n+r)$ permutation matrix. The public key of the scheme is

$$G_{\text{pub}} \overset{\text{def}}{=} S^{-1}G'Q^{-1}.$$  

This cryptosystem can be cryptanalyzed if a description of the GRS code can be recovered from $G_{\text{pub}}$. We give here a way to break this scheme in polynomial time which relies on two ingredients. The first one is given by

Lemma 1 Let $G'$ be a $k \times (n+r)$-matrix obtained by inserting $r$ random columns in a generator matrix of an $[n,k]$ GRS code $C$. Let $C'$ be the corresponding code. Assume that $k < n/2$, then

$$2k - 1 \leq \dim <C'^2> \leq 2k - 1 + r.$$
The proof of this statement is given in Appendix B. Actually the right-hand inequality of Lemma 1 is sharp and with very high probability we observe that

\[ \dim < \mathcal{C}' > = 2k - 1 + r. \]

A discussion which explains this behavior is given in Appendix B. When \( 2k - 1 + r \geq n \), \(< \mathcal{C}'^2 >\) is typically the whole ambient space \( \mathbb{F}_q^n \). This will be useless to detect the positions which correspond to the \( C_i \)'s. We call such positions the \textit{random positions} whereas the other positions are referred to as the \textit{GRS positions}. We use in this case a shortening trick which relies upon the following well-known fact

\textbf{Fact 1.} Shortening a GRS code of parameters \([n, k]\) in \( \ell \leq k \) positions gives a GRS code with parameters \([n - \ell, k - \ell]\).

An attack easily follows from these facts. First of all, let us consider the case when \( 2k - 1 + r \leq n \), then consider \( \mathcal{C}'(i) \) which is the shortened \( \mathcal{C}' \) code at position \( i \). Two cases can occur

- \( i \) belongs to the random positions, then we expect that the dimension of \( \mathcal{C}'(i) \) is given by

\[ \dim < \mathcal{C}'(i)^2 > = 2k - 2 + r. \]

since \( \mathcal{C}'(i) \) is nothing but a \( k \)-dimensional GRS code with \( r - 1 \) random columns added to its generator matrix.

- \( i \) belongs to the GRS positions, then \( \mathcal{C}'(i) \) is a \( k - 1 \)-dimensional GRS code with \( r \) random columns added to its generator matrix and we expect that

\[ \dim < \mathcal{C}'(i)^2 > = 2k - 3 + r. \]

This gives a straightforward way to distinguish between the random positions and the GRS positions.

Consider now the case where \( 2k - 1 + r > n \). The point is to shorten \( \mathcal{C}' \) in \( a \) positions in order to be able to apply again the same principle. Here \( a \) is chosen such that \( a < k \) and \( 2(k - a) - 1 + r < n - a \implies a > 2k - 1 + r - n \). Notice that these conditions on \( a \) can be met as soon as \( k > 2k + r - n \implies n > k + r \). Among these \( a \) positions, \( a_0 \) of them are random positions and \( a_1 \) are GRS positions. This yields a GRS code of parameters \([n - a_1, k - a_1]\) to which \( r - a_0 \) random positions have been added (or more precisely this yields a code with generator matrix given by the generator matrix of a GRS code of size \((k - a_1) \times (n - a_1)\) with \( r - a_0 \) random columns added to it). Denote by \( \mathcal{C}'_a \) this shortened code. Using the previous results, we get that with high probability,

\[ \dim < \mathcal{C}'_a^2 > = 2(k - a_1) - 1 + r - a_0 \]

To identify which positions of \( \mathcal{C}'_a \) are random positions and which ones are GRS positions we just use the previous approach by shortening \( \mathcal{C}'_a \) in an additional position and checking whether or not the dimension decreases by one or two. This approach has been implemented in Magma and leads to identify easily all the random columns for the parameters suggested in [Wie06]. After identifying the random columns in the public generator matrix, it just remains to puncture the public code at these positions and to apply the Sidelnikov-Shestakov attack to completely break the scheme proposed in [Wie06].
6 Conclusion

The homomorphic scheme suggested in [BL11] actually leads in a natural way to choose codes for which the square product is of unusually small dimension (see Appendix ?? for more details). This sheds some light on why considerations of this kind might lead to an attack. It is worthwhile mentioning that replacing Reed-Solomon codes by Reed-Muller ones for instance in this scheme does not seem to prevent this kind of attack.

Both attacks we presented here against [BL11,BBC+11] may be viewed as trying to identify, through square code dimension considerations, certain subcodes or punctured codes of the public codes of the schemes. In the case of Bogdanov-Lee’s scheme, this was for identifying the punctured codes with a certain number of elements of $L$ in their support. In the Baldi et al. case, this was for identifying codewords in a subcode of codimension 1. Reed-Solomon codes are particularly prone to this kind of attack because of the very low dimension of their square code.

The approach we developed here seems to have other applications to cryptanalysis. For instance, it is not too difficult to use it for finding another way of breaking a McEliece type cryptosystem based on generalized Reed-Solomon (the Sidelnikov-Shestakov attack [SS92]) which would start by trying to identify the subcode $\text{GRS}_{k-1}(x,y)$ of the generalized Reed-Solomon code $\text{GRS}_k(x,y)$. It might also be applied to other codes such as for instance Reed-Muller codes [Sid94]. The square code of these codes have also an abnormal dimension. Finally, the most challenging task would be to attack the McEliece cryptosystem with similar tools (at least for a range of parameters) since duals of Goppa codes also have, in a limited way, square codes with low dimensions.$^7$

Acknowledgement: We thank the anonymous reviewer for a careful reading of this submission which helped us to improve its editorial quality.

$^7$ See [MCP12] which contains much more examples of codes with this kind of behavior
A Correctness of the Bogdanov-Lee decoding procedure

Let us explain here why the decryption algorithm outputs the correct plaintext when $\ell$ and $n$ are chosen such that the entry $e_i$ at position $i$ of the error vector is zero when $i \in L$. If this property on $e$ holds, notice that the linear system (1) has $3\ell$ unknowns and $\ell + 1$ equations and since it is by construction of rank $\ell + 1$, it always admits at least one solution. Then observe that

$$\sum_{i=1}^{n} y_i c_i = (xP + m1 + e)y^T$$

$$= (xP + m1)y^T \quad \text{(since } e_i = 0 \text{ if } i \in L \text{ and } y_i = 0 \text{ if } i \notin L \text{)}$$

$$= xSGy^T + m \sum_{i=1}^{n} y_i$$

$$= m \quad \text{(since } Gy^T = 0 \text{ and } \sum_{i=1}^{n} y_i = 1 \text{)}.$$

B Proof of Lemma 1

Let us prove Lemma 1.

Proof. The first inequality comes from the fact that puncturing $C^2$ at the $r$ positions corresponding to the added random columns yields the code $C^2$ which is the square of an $[n, k]$ GRS code and hence an $[n, 2k - 1]$ GRS code. To prove the upper bound, let $D$ be the code with generator matrix $G''$ obtained from $G'$ by replacing the $C_i$'s columns by all-zero columns and let $D'$ be the code with generator matrix $G'''$ obtained by replacing in $G'$ all columns which are not the $C_i$'s by zero columns. Since $G' = G'' + G'''$ we have

$$C' \subset D + D'.$$

Therefore

$$< C'^2 > \subset < (D + D')^2 >$$

$$\subset < D^2 > + < D'^2 > + < D \times D' >$$

$$\subset < D^2 > + < D^2 >$$

where the last inclusion comes from the fact that $< D \times D' >$ is the zero subspace since $D$ and $D'$ have disjoint supports. The right-hand side inequality follows immediately from this, since

$$\dim < D^2 > = 2k - 1 \text{ and } \dim < D'^2 > \leq r.$$

References

[BBC+11] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani. Enhanced public key security for the McEliece cryptosystem. submitted, 2011. See arxiv:1108.2462v2[cs.IT]. After learning that their scheme was attacked by this paper, the authors changed their scheme accordingly and the new version is now arxiv:1108.2462v3[cs.IT].


[BL11] A. Bogdanov and C.H. Lee. Homomorphic encryption from codes. See http://arxiv.org/abs/1111.4301. This paper was accepted for publication in the proceedings of the 44th ACM Symposium on Theory of Computing (STOC). The authors withdrew their paper after they learned that their scheme was threatened, 2011.


Linear Propagation in Efficient Guess-and-Determine Attacks

Maria Eichlseder · Florian Mendel · Tomislav Nad · Vincent Rijmen · Martin Schläffer

Received: date / Accepted: date

Abstract The most successful attacks on cryptographic hash functions are based on differential cryptanalysis, where the main problem is to find a differential characteristic. Finding a differential characteristic is equivalent to solving a system of nonlinear equations. Solving these equations is usually done by a guess-and-determine approach. Recently, automated tools performing a guess-and-determine approach based on the concept of generalized conditions have been used to attack many hash functions. The core part of such tools is the propagation of information. In this paper, we propose a new approach to propagate information for affine functions and compare it to the approach used in recent hash function attacks. We apply our method to the linear functions $\sigma_i$ and $\Sigma_i$ used in SHA-2 and to the linear layer of SHA-3. We show that our approach performs much better than the previously used methods.

Keywords hash functions · differential cryptanalysis · automated tool · guess-and-determine · linear functions · SHA-2 · SHA-3

CR Subject Classification Cryptography

1 Introduction

Finding collisions or preimages for hash functions is a special case of the general problem of solving nonlinear equations. One of the general approaches for solving nonlinear equations is the guess-and-determine approach. Many attacks on cryptographic hash functions can be described as guess-and-determine attacks. Depending on the function, attack setting and other properties, the system of equations can be simplified such that a guess-and-determine approach can be successful. The basic idea of a guess-and-determine approach is to perform a guessing of certain bits before determining others. For instance, message modification and
advanced message modification \cite{14,15} and all their successors and variants perform a guessing of certain bits before determining others.

With the selection of the AES, interest in cryptanalysis using algebraic methods re-awakened, due to the simple algebraic structure of this cipher \cite{3,4}. After the initial optimism, it became clear that although it is possible to construct relatively simple sets of equations, one still needs efficient equation solving strategies. Several attempts were made in that direction, one of the most promising being Multiple Right-Hand Side equations \cite{12,13}.

The most successful attacks on hash functions are based on differential cryptanalysis \cite{2}. The original description of differential cryptanalysis assumes that a good characteristic is given. However, it is often not feasible to find characteristics by hand, in particular for hash functions. Cryptographers now use (semi-)automated tools to find differential characteristics. Based on the concept of generalized conditions, De Cannière and Rechberger provided the first tool to find automated differential characteristics for SHA-1 based on a guess-and-determine approach \cite{5}. Recently, this work was improved and extended, leading to attacks on several hash functions \cite{8–11}. A similar approach is followed by Leurent \cite{6,7}.

The core part of these tools is the propagation of constraints on the variables.

In this paper, we propose a method to store and propagate linear relations of variables efficiently. Since most cryptographic algorithms consist of large linear parts, we show that storing linear relations on more than a single bit leads to significantly better results.

In the following, we start with an introduction to guess-and-determine attacks in differential cryptanalysis in Section 2 and review previous methods and their limitations in Section 3. We continue with the main part of this paper, the linear propagation of linear information in Section 4 and evaluate it in Section 5. Finally, we conclude in Section 6.

2 Guess-and-Determine Attacks

On a high level, a guess-and-determine attack can be described as a repetition of two steps until all unknown variables have been determined: first, guess the value of some unknowns; second, determine the value of as many unknowns as possible. The second step can be slightly generalized. Instead of outputting only the values of new unknowns, it can also output simplified equations between remaining unknowns or partial information (constraints) on the remaining unknowns. It can also announce that for the currently guessed values, there is no solution to the system, signaling that some of the guesses need to be changed. Therefore, we will refer here to the second step using the more general term propagation of information.

A successful guess-and-determine attack employs a strategy to convert complex and dense equations into a form that is more amenable to analyze. The following elements need to be considered in such a strategy.

Choice of intermediate variables: By introducing additional intermediate variables, it becomes possible to reduce the algebraic degree and/or improve the sparsity of the equations. Of course, every newly introduced unknown also introduces a new equation. Hence, there is a trade-off between the number of equations and the simplicity of each equation.
Choice of information to store: Storing all information on each of the intermediate variables would require too much effort to keep all information up-to-date and consistent. Therefore, we store only a part of the information, and recreate the rest if we need it.

Propagation of information: Every time a variable is guessed, we need to check whether new information on other variables can be determined. There is a trade-off between the effort we spend in this step and simply guessing more bits.

Guessing strategy: We need a guessing strategy which can efficiently use the new information generated by the propagation of information introduced by previous guesses.

3 Bitsliced Propagation of Information

In efficient guess-and-determine attacks, we propagate information by solving equations. The complexity and difficulty of this equation solving step depends on several factors as mentioned in the previous section. In the case of cryptographic primitives, these equations are usually large, complex and hard to solve. Therefore, we split these equations into easier parts which can be solved more efficiently.

Recent guess-and-determine attacks on hash functions split the equations into small bitslices and store information on intermediate variables using generalized conditions \[5, 8–11\]. The subproblem of propagating information within bitslices is small enough to be solved efficiently using exhaustive search.

3.1 Generalized Conditions

In classical differential cryptanalysis, only one bit of information is stored for each pair of bits \((x_j, x_j^*)\): the difference \(\Delta x_j = x_j \oplus x_j^*\). Inspired by signed-bit differences \[14\], De Cannière and Rechberger introduced generalized conditions \[5\], where all information on one pair of bits \((x_j, x_j^*)\) is taken into account.

**Definition 1** Let \((x_j, x_j^*)\) be a pair of bits. The **generalized condition** \(\nabla(x_j, x_j^*)\) is a subset of all pairs \(\{(0,0), (0,1), (1,0), (1,1)\}\) of \((x_j, x_j^*)\).

In total, we get 16 possible generalized conditions. We use the same notation as in \[5\] (see Table 1). To specify differences and conditions on word level, we group generalized conditions as follows. Let \(x, x^* \in \{0,1\}^n\) and \(c_j = \nabla(x_j, x_j^*)\). Then, the notation \(\nabla(x, x^*) = [c_{n-1} \ldots c_0]\) provides a compact specification of all \(n\) generalized conditions of the pair \((x, x^*)\). Let \(\nabla(x, x^*) = [c_{n-1} \ldots c_0]\). Then

\[|\nabla(x, x^*)| = \prod_{i=0}^{n-1} |c_i|\]

denotes the number of pairs fulfilling all generalized conditions in \(\nabla(x, x^*)\).
3.2 Bitsliced Propagation

In the bitslice approach, information is propagated only between bits of a bitslice. A bitslice is defined as follows.

Definition 2 Let \( f : \{0, 1\}^m \to \{0, 1\}^n \) be a function with \( y = f(x) \). One output bit \( y_j \) depends on \(|I|\) inputs with \( I \subseteq \{0, \ldots, m - 1\} \). Then the set \( B_j = \{y_j, x_{i_0}, \ldots, x_{i_{|I|}}\} \) is called a bitslice.

In a guess-and-determine attack, new conditions are imposed on bits of a bitslice. These new conditions affect at least the bits in the same bitslice. Hence, the generalized conditions of a bitslice need to be updated. This propagation of new information is done as described in Algorithm 1.

Algorithm 1 Propagation

Input: \( \nabla(x_i, x_{i}^*) \)
Output: Updated conditions on bits in all bitslice pairs \((B_j, B_j^*)\) containing \( x_i, x_{i}^* \)

for each pair of bitslices \((B_j, B_j^*)\) containing \( x_i \) and \( x_{i}^* \) do
  Test all possibilities allowed by the generalized conditions
  Remove cases which are not possible any more

Algorithm 1 provides an optimal propagation for bitwise Boolean functions, where each bitslice is independent of each other. Examples are the bitwise Boolean functions \( IF \) and \( MAJ \) used in many hash function designs of the MD4-family. Furthermore, it has been shown in recent attacks on the MD4-family of hash functions \([5, 8–11]\) that the bitslice approach also works quite well for the modular addition. However, it has significant shortcomings when applied to more complex (linear) functions as shown in the next section.

3.3 Example for Bad Propagation

The shortcomings of the bitsliced approach can be illustrated best by means of an example. We consider the linear \( \Sigma_0 \) function which is used in the hash function SHA-2.

Example 1 Let \( \Sigma_0 : \{0, 1\}^{32} \to \{0, 1\}^{32} \) be a linear function with

\[
\Sigma_0(x) = (x \gg 2) + (x \gg 13) + (x \gg 22) = y,
\]

which is equivalent to \( y_j = x_{j+2} + x_{j+13} + x_{j+22} \), for \( j = 0, \ldots, 31 \). Hence, \( \Sigma_0 \) defines 32 bitslices \( B_j = \{y_j, x_{j+2}, x_{j+13}, x_{j+22}\} \). Note that the additions in the indices are modulo 32. Assume that we start with the following information:

\[
\nabla(x, x^*) = [????????????????????????????????????-??]
\]

\[
\nabla(y, y^*) = [-----------------------------].
\]

By filling in the information for bitslice \( B_0 \) we get

\[
\nabla(x_2, x_2^*) = [-]
\]

\[
\nabla(x_{13}, x_{13}^*) = [-]
\]

\[
\nabla(x_{22}, x_{22}^*) = [?]
\]

\[
\nabla(y_0, y_0^*) = [-].
\]
We can then propagate information according to Algorithm 1 by exhaustively searching over all possible pairs defined by the input generalized conditions, to check which output pairs are possible (see Table 1). It follows that \( \nabla(x_{22}, x_{22}^*) = [-] \). Note that in all the other cases no information is propagated and applying the algorithm to all other indices gives the following propagated information:

\[
\begin{align*}
\nabla(x, x^*) &= [?????????-????????-????????-??] \\
\nabla(y, y^*) &= [-----------------------------].
\end{align*}
\]

However, for the invertible linear function \( \Sigma_0 \) we know that \( \nabla(x_j, x_j^*) = [-] \) must hold for all \( j \).

In this case, the bitslice approach of propagating information performs very poor. Therefore, we present in the following section a different approach which produces optimal results for affine functions and linear conditions.

### 4 Linear Propagation

The constraints defined by generalized conditions and the function \( f \) can also be expressed as a system of equations involving the input and output bits as variables (see Table 1). Propagating conditions then corresponds to manipulating this system in order to bring it to a more useful form. If all involved equations are linear, methods like elementary row and column operations can be applied to simplify the system. Unfortunately, it is not possible to translate all generalized conditions into linear equations. Furthermore, hash functions contain nonlinear building blocks such as modular additions or other nonlinear Boolean functions which cannot be expressed as linear equations.

However, large parts of cryptographic algorithms consist of affine (linear) functions. Furthermore, most generalized conditions are linear (only \( T, B, D, \) and \( E \) are nonlinear). In the following, we show a general method to efficiently extract linear information on the variables and propagate this information.

#### 4.1 Affine Functions

Let \( f : \{0,1\}^m \rightarrow \{0,1\}^n \) be an affine function. Such a function can be described by a matrix \( L \in \{0,1\}^{n \times (m+n)} \), where

\[
y = f(x) \Leftrightarrow L \begin{bmatrix} x \\ y \end{bmatrix} = d
\]

and \( d = f(0) \). When considering input pairs \( x \) and \( x^* \), the same affine function is applied to both inputs. Denoting the vector \( [x \ y]^T \) by \( z \) and \( [x^* \ y^*]^T \) by \( z^* \) respectively, the whole system of linear equations can then be written as follows:

\[
\begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} z \\ z^* \end{bmatrix} = \begin{bmatrix} d \\ d \end{bmatrix}.
\]  (2)
4.2 Linear Conditions

**Definition 3** A linear generalized condition $\nabla L(z_j, z_j^*)$ is a generalized condition that is an affine space.

A set of linear generalized conditions on several bits can be expressed as

$$
\begin{bmatrix}
C \\
C^*
\end{bmatrix}
\begin{bmatrix}
z \\
z^*
\end{bmatrix} = b,
$$

where $C, C^* \in \{0, 1\}^{c \times (m+n)}$ are binary matrices with $c$ linear equations on $m+n$ bits $z, z^* \in \{0, 1\}^{m+n}$ and $b \in \{0, 1\}^c$. Table 1 shows how the matrices $C$ and $C^*$ are constructed from the linear generalized conditions.

### Table 1 Notation and equations of the 16 generalized conditions. For linear conditions, matrix coefficients $C_{ij}, C^*_{ij}$ are given. The elements in rows $C_i$ and $C^*_i$ are zero except for the given positions.

| $\nabla(z_j, z_j^*)$ | (1, 1) | (0, 1) | (1, 0) | (0, 0) | Equations | $C_{ij}$ | $C^*_{ij}$ | $b_i$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>contradiction</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$z_j = 0$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>-</td>
<td>-</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j^* = 0$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>$z_j = 0$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>n</td>
<td>-</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>$z_j^* = 1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j = 0$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>-</td>
<td>$z_j + z_j^* = 0$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>$\times$</td>
<td>no constraints</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4.3 Linear Propagation of Information

In order to propagate linear information, we perform the following three steps:

1. Construct the combination of $[\mathbf{2}]$ and $[\mathbf{3}]$.
2. Use Gauss-Jordan elimination to create sparse equations (rows).
3. Convert equations only on $z_j$ and $z_j^*$ back to generalized conditions.

We combine the two systems $[\mathbf{2}]$ and $[\mathbf{3}]$ such that the resulting matrices $A, A^* \in \{0, 1\}^{(2n+c) \times (m+n)}$ represent the columns corresponding to $z$ and $z^*$ respectively:

$$
\begin{bmatrix}
L & 0 \\
0 & L \\
C & C^*
\end{bmatrix}
\begin{bmatrix}
z \\
z^*
\end{bmatrix} =
\begin{bmatrix}
A & A^*
\end{bmatrix}
\begin{bmatrix}
z \\
z^*
\end{bmatrix} =
\begin{bmatrix}
d \\
b
\end{bmatrix}.
$$

(4)
To propagate linear information we simply perform Gauss-Jordan elimination and get a matrix in reduced row-echelon form. If the system is inconsistent, we know that the generalized conditions at the input and output of the affine function contradict each other. If the system is consistent, we can extract new information in form of generalized conditions. Since a generalized condition consists of equations involving only $z_j$ and $z_j^*$, we get this information by a linear combination of at most two rows.

5 Comparison of Propagation Methods

In this section, we compare the different information propagation methods. We show that in the majority of cases the linear propagation described in Section 4.3 performs much better than the bitslice approach described in Section 3.2. For the rare cases where the bitslice approach performs better, a combination of both approaches results in the best propagation performance. To compare and evaluate the different propagation methods, we need to measure how well they propagate. Propagation corresponds to narrowing down the solution space, or gaining information about the solution.

Let $\nabla(z, z^*)'$ denote the generalized conditions obtained by propagating $\nabla(z, z^*)$ by means of propagation method $M$. Then we take as figure of merit for $M$:

\[
I_M(z) = \log_2 \frac{|\nabla(z, z^*)|}{|\nabla(z, z^*)'|}
\]

If $\nabla(z, z^*)$ is a contradiction, then $|\nabla(z, z^*)| = |\nabla(z, z^*)'| = 0$ and we set $I_M(z) = 0$. If $|\nabla(z, z^*)'| = 0$ but $|\nabla(z, z^*)| \neq 0$, then $I_M(z)$ is undefined, which we denote by $I_M(z) = \#$. To compare the two propagation methods and measure the gain of the linear method (L) over the bitslice method (B) for one specific condition $\nabla(z, z^*)$, we compute the difference of the two methods’ figures of merit:

\[
I_{\text{diff}}(z) = I_B(z) - I_L(z).
\]

If $I_L(z) = \#$ but $I_B(z) \neq \#$, the linear method detects the contradiction but the bitslice method does not. In this case, we set $I_{\text{diff}}(z) = \#_L$. If $I_L(z) \neq \#$ but $I_B(z) = \#$ we set $I_{\text{diff}}(z) = \#_B$. If both are $\#$, we set $I_{\text{diff}}(z) = 0$.

5.1 Applications

We evaluate the propagation methods for different functions $f$ by computing $I_{\text{diff}}(z)$ for a large number of randomly drawn samples $\nabla(z, z^*)$. Choosing random generalized conditions at the input and output of one of the functions results in impossible characteristics with a high probability. Since such cases are less likely to occur in a guess-and-determine attack, we have performed a search for differential characteristics in SHA-2 and Keccak (SHA-3) and extracted random samples from this search. The results show the empiric distribution function of the random variable $I_{\text{diff}}(z)$. 
5.1.1 Linear Functions of SHA-2.

Figure 1 shows the results for the 32-bit (SHA-256) and 64-bit (SHA-512) linear functions $\Sigma_0$ and $\sigma_0$ of SHA-2. The figure shows a comparison for samples, which commonly occur in a search for differential characteristics in SHA-2. The linear propagation performs significantly better than the bitslice approach for $\sigma_0$ which is used in the message expansion of SHA-2. Since $\Sigma_0$ is used in the state update transformation where other functions influence the propagation as well, the gain is limited.

5.1.2 Linear Layer of Keccak.

Finally, we apply the different propagation methods to the linear layer used in Keccak [1] which is significantly larger than the linear functions used in SHA-2. In more detail, the linear layer in Keccak updates 25 lanes, each of length $w$. Since in Keccak lane lengths of 8, 16, 32 and 64 bits are defined, we can compare the propagation methods for different sizes of the linear layer. Using Keccak we can show, that for larger affine functions the linear propagation method tends to perform better. Note that for SHA-3 only a length of 64 bits is defined.

Figure 2 shows the comparison of the bitslice approach with the linear approach for different lane lengths in Keccak. Most notably, for $w = 64$, the linear approach performs better in more than 97% of the samples, and significantly more contradictions are detected by the linear approach.
Linear Propagation in Efficient Guess-and-Determine Attacks

Fig. 2 Comparison of propagation methods for different lane lengths ($w = \{8, 16, 32, 64\}$) of the whole linear layer used in Keccak. Higher values correspond to better propagation.

6 Conclusions

In recent years, many differential attacks on hash functions are using automated tools to find differential characteristics. These tools are performing a guess-and-determine attack to solve the resulting nonlinear equations. The basic idea of a guess-and-determine attack is to perform a guessing of certain bits before determining others. The core part of such tools is the propagation of (new) information. Usually, a trade-off between the performance of the propagation and the amount of propagated information is made. Hence, information is lost and a better propagation method can significantly improve these tools.

We investigated how the propagation for certain functions can be improved. We focused on affine functions, since most cryptographic algorithms consist of large linear parts. We showed that the approach used in the recent attacks has some significant drawbacks. As a solution to this problem, we proposed a linear information propagation method and showed that it performs significantly better.

In our approach, we store linear relations on more than a single bit. Furthermore, instead of propagating information only within a single bitslice, we propagate information wordwise by considering the whole system of linear equations derived from the affine function and generalized conditions. Using an efficient algorithm to extract new information from this system, we achieve optimal results for linear functions and linear generalized conditions. Furthermore, we gave a detailed description of the approach and compared it to the approach used in the recent hash function attacks. For the comparison, we applied our method to the linear functions $\Sigma_0$ and $\sigma_0$ used in SHA-2 and to the linear layer of Keccak (SHA-3). We have shown that our method performs much better than previously published methods, making it possible to apply the recent techniques to more complex functions like SHA-3.
Acknowledgments

Part of this work was done while Florian Mendel was with KU Leuven. The work has been supported by the Austrian Science Fund (FWF), project P21936-N23 and TRP 251-N23; and by the Research Fund KU Leuven, OT/08/027.

References

A New Algorithm for Finding Low-Weight Polynomial Multiples and its Application to TCHo

Thomas Johansson · Carl Löndahl

Abstract In this paper we present an algorithm for finding low-weight multiples of polynomials over the binary field using coding theoretic methods. The code defined by the public polynomial is cyclic, allowing an attacker to search for any shift of the sought codeword. Therefore, a code with higher length and dimension is used, having a larger number of low-weight codewords. Additionally, since the degree of the sought polynomial is known, the sought codewords of weight w are transformed by a linear mapping into codewords of weight w − 2. Applying an algorithm for finding low-weight codewords on the constructed code yields complexity for a key-recovery attack against TCHo that is lower than previously expected.

Keywords Low-weight polynomial multiple, low-weight codeword, information-set decoding, public-key cryptography, TCHo

1 Introduction

In public-key cryptography there are several different cryptosystems based on different hard problems. Integer factorization and discrete logarithms are among the most common and most trusted problems used. But many other problems, like knapsack problems, hard coding theory problems, lattice reduction problems etc. are also intensively studied. One motivation for this is that a large future quantum computer, if ever constructed, will be able to solve factorization and discrete logarithms in polynomial time [20], whereas other problems might remain more difficult. For a more detailed discussion on this, see [5]. A second motivation is that in constrained environments we might see a sufficient performance gain when we use some alternative cryptosystem compared to common ones (like RSA, El Gamal) based on factorization or discrete logarithms.

A very old such alternative cryptosystem is the McEliece public-key cryptosystem from 1978 [18], based on the difficulty of decoding a general random code. Still, it
remains unbroken, although advances in algorithms for solving the decoding problem have forced parameters to be larger than originally proposed.

In a much more recent, but similar direction, Finiasz and Vaudenay proposed in 2006 a public-key cryptosystem called TCHo [11], based on the problem of finding low-weight multiples of a given binary polynomial. The original proposal was refined in 2007 by Aumasson, Finiasz, Meier and Vaudenay in [3]. Herrmann and Leander demonstrated in [14] a chosen-ciphertext attack on the basic form of TCHo, implying that the use of TCHo with a hybrid encryption framework as originally proposed in [11] is the way to use TCHo.

Finding a low-weight multiple $K(x)$ of a binary polynomial $P(x)$ is believed to be a computationally hard problem. As of today, no known polynomial-time algorithm exists. Although it is not known how hard the problem is, one can easily show that it is not harder than finding a low-weight codeword in a random code. For the latter problem, Stern’s algorithm [15] can be used. It is a probabilistic algorithm with an expected exponential running time. Throughout the years, Stern’s algorithm has undergone several improvements [7,12,6,17,16,4].

The problem of finding a low-weight multiple is of great importance in general in the field of cryptography. Some other applications are distinguishing attacks and correlation attacks on stream ciphers [19,8,1]. Another area is finite field arithmetics. The results we present in this paper apply equally well to these areas and improve state-of-art also here, even though we do not explicitly give the details.

We present a new algorithm for finding low-weight polynomial multiples of a binary polynomial. The algorithm has a lower complexity than existing approaches for solving this problem. We focus on the actual computational complexity of the algorithm and not the asymptotics, similar to previous work in this area [17,16,4].

The algorithm is then used to attack the public-key cryptosystem TCHo [3]. A consequence of the new algorithm is that the gap between claimed security level and actual algorithmic complexity of a key recovery attack on TCHo is narrowed. For some parameters, the complexity might be interpreted as below the security level.

The paper is organized as follows. In Section 2, we shortly describe the LWPM problem and a few variations of it. In Section 3, a description of the new algorithm is given, followed by a complexity analysis in Section 4. In Section 5, we apply the algorithm to the TCHo cipher. Section 6 concludes the paper.

2 The Low-Weight Polynomial Multiple Problem

**Definition 1** The weight (or Hamming-weight) of a binary vector $v = [v_0 \, v_1 \ldots v_{n-1}]$ is the number of non-zero elements, i.e., $w_H(v) = |\{i : v_i \neq 0, \ 0 \leq i < n\}|$, while the weight of a polynomial $P(x) = p_0 + p_1x + \cdots + p_{n-1}x^{n-1}$ is the number of non-zero coefficients, $w_H(P(x)) = |\{i : p_i \neq 0, \ 0 \leq i < n\}|$.

Let us start by defining the problem of finding all low-weight polynomial multiples as follows:

**Problem 1** All Low–Weight Polynomial Multiples (ALWPM)

**Input:** A polynomial $P(x) \in \mathbb{F}_2[x]$ of degree $d_P$. Two integers $w$ and $d$.

**Output:** All multiples $K(x) = P(x)Q(x)$ of weight at most $w$ and degree at most $d$.

A related but different problem emerges when it is sufficient to find one single solution among possibly several solutions, i.e.,
Problem 2 Low–Weight Polynomial Multiple (LWPM\textsubscript{A})

**Input:** A polynomial $P(x) \in \mathbb{F}_2[x]$ of degree $d_P$. Two integers $w$ and $d$.

**Output:** One (if it exists) multiple $K(x) = P(x)Q(x)$ of weight at most $w$ and degree at most $d$.

A major difference between these two problems lies in the fact that generalized birthday arguments [21] can sometimes be used in Problem 2 whereas it is usually not applicable to Problem 1, as this technique does not necessarily find all possible solutions. It is also a question of the nature of the underlying problem giving rise to the LWPM problem. In some cases a single multiple is enough and in other cases one is interested in finding many. We can also imagine a problem formulation where we output $T$ multiples instead of all.

Another deviating point is the expected number of multiples. A rough estimation gives the expected number of low-weight multiples to be around $\frac{d^w - 1}{(w-1)!2^{d_P}}$. We may then have a set of instances where the expected number of low-weight multiples is very low, but we know from construction that such a multiple does exist. The second scenario is when the expected number of multiples is larger and the problem instance is perhaps a random instance ($P(x) \in \mathbb{F}_2[x]$ of degree $d_P$ is randomly chosen among all such polynomials).

Looking at known algorithms for solving the LWPM problem, the techniques differ depending on whether we are considering a fixed very small weight $w$, typically $w = 3, 4, 5$, or whether we are considering larger values of the weight $w$.

Undoubtedly, there are many flavours of this problem. We consider here the case relevant to TCHo. Thus, we give another modified problem formulation that fits the TCHo case.

Problem 3 Low–Weight Polynomial Multiple (LWPM\textsubscript{B})

**Input:** A polynomial $P(x) \in \mathbb{F}_2[x]$ of degree $d_P$. Two integers $w$ and $d$.

**Output:** One (if it exists) multiple $K(x) = P(x)Q(x)$ of weight exactly $w$ and degree exactly $d$.

Let us give a very brief overview of previous algorithms for these problems. Several algorithms have a large initial cost. Thus, for some parameters an exhaustive search will have the best complexity.

### 2.1 Time–memory trade-off approach

There exists a plethora of variations and improvements of this method. Among these, we find for instance the approach by Golić [13]. The algorithm formulated by Golić searches for polynomials of weights $w = 2^j$ (and $w = 2^j - 1$). The initial step consists of creating a list that contains the $\binom{n}{j}$ residues of the form $x^{i_1} + x^{i_2} + \cdots + x^{i_j} \mod P(x)$, for $0 \leq i_1 < i_2 < \cdots < i_j < n$. These residues can be efficiently represented as integers, on which it is straightforward to apply a sort-and-match procedure. Any collision gives rise to a polynomial of weight $2^j$ being a multiple of $P(x)$. The algorithm requires time and memory of about $\binom{n}{j}$.

Another approach is the match-and-sort approach by Chose et al. [9]. Using a divide-and-conquer technique, the task of finding collisions in a search space of size $n^w$, is divided into smaller tasks. This is done by searching for collisions in smaller subsets with less restrictions. The solutions to the smaller subproblems are then sorted and put together to solve the whole problem. This approach has time complexity of about $n^{\lceil w/2 \rceil} \log n$ and requires $n^{\lfloor (w-1)/4 \rfloor}$ of space.
In [10], Didier and Laigle-Chapuy consider using discrete logarithms instead of the direct representation of the involved polynomials to improve performance. When the degree of the multiple can be large and there are many low-weight multiples, but it is sufficient to find only one, Wagner’s generalized birthday approach [21] becomes efficient.

2.2 Finding minimum-weight words in a linear code

The low-weight polynomial multiple problem can be reduced to the problem of finding a low-weight codeword in a linear code $C$. An $(n,k)$ linear code is a linear subspace of dimension $k$ of a vector space $\mathbb{F}_q^n$, defined by $C \triangleq \{ uG : u \in \mathbb{F}_q^k \}$ where $G$ is a $k \times n$ matrix over $\mathbb{F}_q$, called the generator matrix. The problem is to find a vector $[v_0 \ v_1 \ \cdots \ v_{n-1}] = v \in C \setminus \{0\}$ such that $v$ has a low weight $w$. This problem is commonly known as the function problem version of Subspace Weight and it is NP-hard. In our case $\mathbb{F}_q = \mathbb{F}_2$.

There are known algorithms for this problem, so-called information-set decoding algorithms. Among these, we find Stern’s algorithm and the improved algorithm by Canteaut and Chabaud [7]. The more recent improvements [17,16,4] also apply.

A common technique to reduce the LWPM-problem into the function problem version of Subspace Weight is the following. Let

$$P(x) = p_0 + p_1 x + \cdots + p_{d_P} x^{d_P}$$

be the given polynomial and let

$$u = [u_0 \ u_1 \ \cdots \ u_{d-d_P}]$$

be a length $d - d_P + 1$ binary vector. One can formulate the problem of finding a weight $w$ polynomial $K(x)$ of degree $\leq d$, being a low-weight multiple of $P(x)$, as finding a binary vector $u$ such that $uG(x)$ has exactly $w$ non-zero coefficients, where

$$G(x) = \begin{bmatrix} P(x) \\ xP(x) \\ \vdots \\ x^{d-d_P} P(x) \end{bmatrix}.$$ 

Writing also the polynomials as length $d$ vectors, the problem reduces to finding a weight $w$ codeword in the linear code generated by the Toeplitz generator matrix

$$G = \begin{bmatrix} p_0 & p_1 & \cdots & p_{d_P} \\ p_0 & p_1 & \cdots & p_{d_P} \\ \vdots & \vdots & \ddots & \iddots \\ p_0 & p_1 & \cdots & p_{d_P} \end{bmatrix}.$$  

(1)

having dimension $(d - d_P + 1) \times (d + 1)$, where the empty spaces correspond to zero-elements. As the problem is reduced to finding a weight $w$ codeword in the linear code generated by $G$, conventional information-set decoding algorithms can be used.
3 The new algorithm solving LWPMb

Let us describe the main ingredients in the new algorithm. We use the established technique from coding theory described in Subsection 2.2, but we introduce some new modifications.

Recall that \( K(x) \) represents the low-weight polynomial multiple we are looking for. Having the generator matrix \( G \) described in (1) in mind, our initial observation is that one can increase success probability in each iteration of information-set decoding part by a factor \( y + 1 \) by allowing \( y \) shifts of the polynomial \( K(x) \), i.e., including the polynomial multiples \( xK(x), x^2K(x), \ldots, x^yK(x) \) along with \( K(x) \) in the solution space (this will be proved in Section 4). The trade-off is that the dimension of the generator matrix grows to \( (d - d_P + 1 + y) \times (d + 1 + y) \). The new generator matrix will have the following structure:

\[
G_y = \begin{bmatrix}
p_0 & p_1 & \cdots & p_{d_P} \\
p_0 & p_1 & \cdots & p_{d_P} \\
\vdots & \vdots & \ddots & \vdots \\
p_0 & p_1 & \cdots & p_{d_P} \\
p_0 & p_1 & \cdots & p_{d_P} \\
\vdots & \vdots & \ddots & \vdots \\
p_0 & p_1 & \cdots & p_{d_P}
\end{bmatrix}
\]  

(2)

The gray rectangle represents \( G \) and everything outside represents the expansion. Let the expanded matrix in (2) be denoted \( G_y \) and the code it generates be denoted \( C_y \).

Recall that the unknown low-weight polynomial is written in the form \( K(x) = 1 + k_1x + \cdots + k_{d-1}x^{d-1} + x^d \), i.e. the polynomial \( K(x) \) has degree \( d \). We will now show how to exploit the form of the polynomial.

**Theorem 1** For any polynomial \( P(x) \), there exists a linear map \( \Gamma \) that transforms the code \( C_y \) into a new code given by \( G_y \Gamma \), such that all weight \( w \) codewords corresponding to shifts of \( K(x) \) will have weight \( w - 2 \) in the new code.

**Proof** Given that \( K(x) \) has degree \( d \), its constant term and the coefficient of \( x^d \) are non-zero. Combining the corresponding columns in \( G_y \), i.e., adding the \((d + 1)\)th column to the first column of \( G_y \) and then removing the \((d + 1)\)th column from \( G_y \), will cause the codeword corresponding to \( K(x) \) to decrease by two in weight. The new codeword stemming from \( K(x) \) will have weight \( w - 2 \).

Note that the symbols in the other weight \( w \) codewords will be permuted, but the weight of these codewords stays the same. We can repeat this for the second column, by adding the \((d + 2)\)th column and so on up to the \( y \)th and \((d + y)\)th column (see Fig. 1). The consequence of the approach just outlined is that all codewords that correspond to shifts of \( K(x) \) will have weight \( w - 2 \). It is easy to see that the operations described above can be expressed as a right multiplication by a matrix \( \Gamma \), giving the new generator matrix \( G_y \Gamma \).

\( \square \)
The matrix product $G_y \Gamma$ forms a new generator matrix of dimension $(d - d_P + 1 + y) \times (d + 1)$, as illustrated in Fig. 1. The final step is to apply information-set decoding on the code formed by $G_y \Gamma$, looking for codewords of weight $w - 2$.

For the information-set decoding step, we use an algorithm first presented in [16], given in Algorithm 1. We use this algorithm because it performs better than Stern’s algorithm. The explicit complexity expressions are given in [16]. The algorithm for solving Problem 3 can be summarized as given in Algorithm 2.

4 Complexity

In order to estimate the complexity of the algorithm solving LWPMb, we use previously established results given in [16]. Let $C^*(n, k, w)$ denote the expected number of binary operations performed by Algorithm 1 to find a codeword of weight $w$ in an $(n,k)$-code $C$, assuming that exactly one such codeword exists.

**Theorem 2 (Johansson & Löndahl)**

$$C^*(n, k, w) = \min_{p, z, l} \frac{\binom{n}{w} \mathcal{N} \binom{n-k}{w-k+1} \binom{k}{z+1} 2^{-z-p} t \cdot 2^{t-1} (n-k)}{\mathcal{N} \binom{n-k-z}{w-2p} \left(1 - \left(1 - 2^{-z} \binom{k}{z}\right)^{2} \right)},$$

where $p$, $z$, and $l$ are algorithm parameters.

**Proof** See [16] for a detailed proof.

For an expanded $(n, k)$ code $C_y$ (according to (2)) with weight reduction by $\Gamma$, we have $(n,k) = (d + 1, d - d_P + 1 + y)$. Running Algorithm 2 on $G_y \Gamma$ will require complexity according to the following theorem.
**Algorithm 1** Improved Stern

**Input:** Generator matrix $G$, parameters $p, z, l$

**Output:** A codeword of weight $w$

1. Let $G' = \pi(G)$, where $\pi$ is a random column permutation and $G$ is the generator matrix with $k$ rows and $n$ columns.
2. Bring the generator matrix $G'$ to systematic form: $[I \ Z \ \mathbf{L} \ \mathbf{J}]$, where $I$ is the $k \times k$ identity matrix, $Z$ is a $k \times z$ matrix, $\mathbf{L}$ is a $k \times l$ matrix and $\mathbf{J}$ is a $k \times n - k - z - l$ matrix. Let $\phi(x)$ be a linear map $\phi : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^k$. With $x = [x_0 \ x_1 \ \ldots \ x_{n-1}]$, then $\phi(x) = [x_k \ x_{k+1} \ \ldots \ x_{k+z-1}]$. Likewise, we define $\phi(x) : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^k$ such that with $\phi(x) = [x_k \ x_{k+1} \ x_{k+z+1} \ \ldots \ x_{k+z+l-1}]$.
3. Let $u$ run through all weight $p$ vectors of length $k$. Store all vectors $x = uG'$ such that $\phi(x) = [0 \ 0 \ \ldots \ 0]$ in a sorted list $H_1$, sorted according to $\phi(x)$. This is done by constructing a list $H_0$ containing all vectors $x = uG'$ where $u$ runs through all weight $p/2$ vectors. Then add all pairs of vectors $x, x' \in H_0$ in the list with $\phi(x) = \phi(x')$ and such that the largest index of the nonzero entries in $x$ is smaller than the smallest index of nonzero entries in $x'$.
4. As in previous step, combine the list $H_1$ with itself to receive a new list $H_2$ of all codewords $x = uG'$ with $u$ of weight $2p$, such that $\phi(x) = [0 \ 0 \ \ldots \ 0]$.
5. For each $x \in H_2$, check if the weight of $x$ is $w - 2p$. If no such codeword is found, go to 1.

**Algorithm 2** Solve-LWPMb

**Input:** Polynomial $P(x)$, weight $w$ and parameter $y$

**Output:** A polynomial multiple $K(x)$ of weight $w$

1. From $P(x)$, create the corresponding generator matrix $G$ according to (1).
2. Expand $G$ by $y$ extra entries, yielding in total $y + 1$ codewords that represent $K(x)$, all of weight $w$. Let the expansion be $G_y$.
3. Transform the codewords that represent $K(x)$ to weight $w - 2$, by forming the generator matrix $G_y \Gamma$, in agreement with Theorem 1.
4. Input $G_y \Gamma$ into Algorithm 1, using optimum parameters with respect to (3), to find one codeword $u$ among the $y + 1$ weight $w$ codewords that represent $K(x)$.
5. From $u$, construct $K(x)$ by exhaustive search over at most $y + 1$ polynomials and output $K(x)$.

**Proposition 1** Algorithm 2 has an expected complexity given by

$$\min_{y \geq 0} \frac{C^* (d + 1, d - dp + 1 + y, w - 2)}{y + 1}$$

(3)

when the success probability of one iteration of Algorithm 1 is small.

**Proof (Proof Sketch)** The complexity function $C^*$ refers to the expected complexity of running Algorithm 1 with an instance where we have one single solution, i.e., only one codeword of the weight $w$ exists in the code, whereas in the case of LWPMb, there will exist several weight-$w$ codewords. Having $y + 1$ possible solutions instead of one suggests that finding at least one is $y + 1$ times more likely. However, for this to be true, the probability $\xi$ of finding one single codeword in one iteration must be small (i). In particular, we require that $y\xi < 1$. 

Secondly, we need the events of finding different codewords to be independent of each other (ii). Consider the following. Let the set of shifts of \( K(x) \) represented as vectors be the rows of the matrix

\[
K = \begin{bmatrix}
1 & k_1 & k_2 & \cdots & k_{d-1} & 1 \\
1 & k_1 & k_2 & \cdots & k_{d-1} & 1 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & k_1 & k_2 & \cdots & k_{d-1} & 1
\end{bmatrix}.
\]

The weight reduction of \( G_y \) by \( \Gamma \), will result in a new matrix

\[
K\Gamma = \begin{bmatrix}
0 & k_1 & k_2 & \cdots & k_{d-1} \\
k_{d-1} & 0 & k_1 & k_2 & \cdots & k_{d-2} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
k_{d-y} & k_{d-1} & 0 & k_1 & k_2 & \cdots & k_{d-y-1}
\end{bmatrix}.
\]

Permutation of \( G_y\Gamma \) permutes all codewords accordingly, thus permuting the matrix \( K \) as well. So we have that

\[
\pi(K\Gamma) = \begin{bmatrix}
k_{i_1} & k_{i_2} & \cdots & k_{i_{d_P}} \\
k_{i_1-1} & k_{i_2-1} & \cdots & k_{i_{k-1}} \\
\vdots & \ddots & \ddots & \ddots \\
k_{i_1-y} & k_{i_2-y} & \cdots & k_{i_{d_P-y-1}}
\end{bmatrix}.
\]

For a row to be considered as a possible solution by Algorithm 1, it can have at most \( 2p \) non-zero elements in the first \( d_P \) columns. If all elements in the first \( d_P \) columns of \( \pi(K\Gamma) \) are different, all codewords are independent. For different parameters \( d_P, d_K \) and \( y \), the overlap will vary.

Given (i) and (ii), we can conclude that the probability of finding at least one out of \( y+1 \) codewords is \( 1 - (1 - \xi)^{y+1} \approx (y+1)\xi \), since all codewords are equally likely to be found. Since \( C^* \) is \( O(\xi^{-1}) \), this concludes the proof.

4.1 Simulation results

We consider a toy example of LWPMb, running Algorithm 2 on an instance with a polynomial

\[
P(x) = 1 + x^1 + x^3 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{16} + x^{20} + x^{25} + x^{28} + x^{29} + x^{32} + x^{33} + x^{34} + x^{37} + x^{38} + x^{39}.
\]

We are seeking a weight \( w = 8 \) multiple of \( P(x) \) of degree 62. The solution is

\[
K(x) = 1 + x + x^2 + x^4 + x^{11} + x^{36} + x^{37} + x^{62}.
\]

In Fig. 2, we are plotting the simulated success rate of each iteration of Algorithm 1 as a function of codeword multiplicity \( y+1 \). The solid line shows the theoretical success probability function, according to (3). The triangle-shaped points show the simulated success probability. The square-shaped points show the the simulated success probability of a single iteration when using the weight-reduction technique described.
in Theorem 1. Looking at Fig. 2, we note that the probability increases by the factor $y+1$. We also note that initially the lines are almost linear, but bend as the probability converges to 1.

In Fig. 3, we are plotting the simulated number of operations in one iteration of Algorithm 1 (squared-shaped points). We note that the simulated operation count follows the same curve as the theoretical expected operation count (solid line).

5 The TCHo cipher and its security

TCHo is a public-key cryptosystem based on the LWPMb problem, see [3,11] for more details. The operation of all variants of the public-key cryptosystem TCHo is non-deterministic and can be thought of as transmitting data over a noisy channel. A high level description is as follows. A linear code encodes the message into a codeword, while an LFSR with a random starting state produces a bitstream that together with a low-weight error vector constitutes the noise on the channel. The connection polynomial of the LFSR is the only component of the public key.

The trapdoor of the cipher works in the following way: the secret low-weight polynomial of fixed degree, which is the only component of the secret key, is divisible by the characteristic polynomial used by the LFSR. Therefore, by “multiplying” the ciphertext with the secret polynomial, the contribution from the LFSR sequence diminishes, leaving only a low-weight error vector as noise. This sequence can then be decoded using conventional decoding algorithms.

One implementation of TCHo is as follows. Let $G_{\text{rep}}$ be a generator matrix of a repetition code of length $l$. The plaintext $m \in \mathbb{F}_2^{128}$ is repeated and the result is truncated to fit a length $l$. These two steps can be represented by multiplication with $G_{\text{rep}}$.

In encoding, let $r = [r_0, r_1, \ldots, r_{l-1}]$ be a random string of $l$ independent bits with bias $\Pr [r_i = 0] = \frac{1}{2}(1 + \gamma)$ (called $\gamma$-biased). Here, $r$ is chosen such that it is highly biased, having a lot more zeroes than ones. Using an LFSR with characteristic polynomial $P(x)$ and a randomly chosen starting state, the LFSR sequence $p$ is generated and truncated to $l$ bits. The encryption is done by adding the three vectors to form the ciphertext $c \in \mathbb{F}_q^l$, where $c = mG_{\text{rep}} + r + p$. 
In encoding, given the secret low-weight polynomial $K(x)$ of fixed degree, we can construct the matrix

$$M = \begin{bmatrix}
k_0 & k_1 & \cdots & k_{dK} 
k_0 & k_1 & \cdots & k_{dK} 
\vdots & \vdots & \ddots & \vdots 
k_0 & k_1 & \cdots & k_{dK}
\end{bmatrix}.$$  

For the decoding step, consider the product $t = cM^T$. From the encryption step we have that

$$t = (mG_{\text{rep}} + r + p)M^T = mG_{\text{rep}}M^T + rM^T + pM^T = mG_{\text{rep}}M^T + rM^T.$$  

Since $P(x)$ divides $K(x)$, we have that $pM^T = 0$. Recall that each bit element in $r$ was $\gamma$-biased. $K(x)$ has weight $w$ and consequently, each element in $r$ will be a sum of $w$ variables that each have bias $\gamma$. Therefore, each element in $rM^T$ will be $\gamma^w$-biased. Here, majority decision decoding can be used to decode

$$t = m(G_{\text{rep}}M^T) + rM^T,$$

i.e., to find a solution $m$ such that the residual $rM^T$ is minimized.

The security of TCHo relies on the hardness of finding the secret polynomial multiple $K(x)$, given only the public polynomial $P(x)$. It is clear that solving LWPM for the instance $P(x)$ would result in a key recovery attack. In [3] and [2], some methods are proposed to solve the LWPM problem. These methods are, however, unable to break TCHo.

5.1 The attack using Algorithm 2

As mentioned, in order to break TCHo we need to find the secret key polynomial $K(x)$ being a low-weight multiple of the public key $P(x)$.

Running Algorithm 2 on the instances proposed in [3], we get the complexities presented in Table 5.1. According to [3], these instances are designed for a security level of $2^{80}$. Note that the other algorithms mentioned in Section 2 have much higher complexity.

<table>
<thead>
<tr>
<th>Instance parameters</th>
<th>Complexity</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_K$</td>
<td>$d_P$</td>
<td>$w$</td>
</tr>
<tr>
<td>25820</td>
<td>7000</td>
<td>45</td>
</tr>
<tr>
<td>24730</td>
<td>12470</td>
<td>67</td>
</tr>
<tr>
<td>44677</td>
<td>4433</td>
<td>25</td>
</tr>
<tr>
<td>24500</td>
<td>8000</td>
<td>51</td>
</tr>
<tr>
<td>17600</td>
<td>8795</td>
<td>81</td>
</tr>
<tr>
<td>31500</td>
<td>13200</td>
<td>65</td>
</tr>
</tbody>
</table>
From a more technical perspective, one can consider the complexity in word operations instead of single bit operations. This would decrease all the complexities above by approximately a factor \(2^6\) and would give a rough estimate of the required number of clock cycles. In implementation, we managed to perform on average \(2^{3.3}\) bit operations per clock cycle.

**Example 1** Consider the case \(d_K = 44677\), \(d_P = 4433\) and \(w = 25\). By minimization of (3) over \(y, p, l\) and \(z\), we obtain the values \(y_{\text{opt}} = 250\), \(p_{\text{opt}} = 4\), \(l_{\text{opt}} = 51\) and \(z_{\text{opt}} = 18\). The generator matrix \(G_y\Gamma\) now has dimension \(40495 \times 44678\) and 251 codewords of weight 23. Using Algorithm 2, we get a complexity of about \(2^{84}\) single bit operations. Assuming that we can perform a majority of the work done by the algorithm using 64-bit word operations, we get a complexity of about \(2^{78}\) word operations.

### 6 Conclusion

We have proposed a new algorithm for finding low-weight multiples of polynomials over \(\mathbb{F}_2[x]\), further improving the complexity for solving LWPMin. And as a consequence, we have narrowed the gap between claimed complexity and actual algorithmic complexity of a key recovery attack on TCHo. If we interpret a claimed security level of \(2^{80}\) as the same number of clock cycles, we have demonstrated at least two proposed instances that have an actual security level below \(2^{80}\).

### Acknowledgements

We would like to thank the anonymous reviewers for their valuable and insightful comments that helped improve the manuscript. This research was funded by a grant (621-2009-4646) from the Swedish Research Council.

### References

Small Secret Exponent Attack on RSA Variant with Modulus
\( N = p^2 q \)

Santanu Sarkar

Abstract We consider an RSA variant with Modulus \( N = p^2 q \). This variant is known as Prime Power RSA. In PKC 2004 May proved when decryption exponent \( d < N^{0.22} \), one can factor \( N \) in polynomial time. In this paper, we improve this bound upto \( N^{0.395} \). We provide detailed experimental results to justify our claim. Keywords: Lattice, Modular Equation, Prime Power RSA

1 Introduction

The famous RSA public key cryptosystem has been proposed in [19] and this is possibly the most studied topic in cryptology research. Though RSA is quite safe till date if applied with proper cryptographic practices, the literature related to its cryptanalysis is quite rich.

In Crypto 1985, Håstad [9] pointed out that short public exponents may cause weakness if the same message is broadcast to many parties. One very important result regarding RSA has been presented by Wiener [22]. He showed that \( N \) can be factored from the knowledge of \( N, e \) if \( d < \frac{1}{4}N^{\frac{1}{2}} \). Latter Boneh and Durfee [2, 3] improved this bound upto \( d < N^{0.292} \) using the seminal idea of Coppersmith [5]. An outstanding survey on the attacks on RSA before the year 2000 is available in [1]. For very recent results on RSA, one may refer to [13, 17] and the references therein.

Apart from the basic RSA proposal there are several variants of it for efficiency and security purposes. In this paper we concentrate on one variant, namely Prime Power RSA. In Prime Power RSA, RSA modulus \( N \) is of the form \( N = p^r q \) where \( r \geq 2 \). In [8], authors presented an electronic cash scheme using the modulus \( N = p^2 q \). In Eurocrypt 1998, Okamoto et al. [18] designed a public key crypto system that is provably as secure as factoring a modulus \( N = p^2 q \). Takagi in [21] introduced the CRT variant of RSA, where the RSA modulus of the form \( N = p^r q \). Boneh et al [4] cryptanalyzed Prime Power RSA when few bits of \( p \) are known. They proved that \( \frac{1}{r+1} \) fraction of MSBs of \( p \) are sufficient to factor \( N \) in polynomial time.
There are two variants of Prime Power RSA. In the first variant $ed \equiv 1 \mod p^{r-1}(p-1)(q-1)$. In the second variant $ed \equiv 1 \mod (p-1)(q-1)$. In [11, 12], authors proved that the second variant is vulnerable when $d < N^{\frac{1}{3\sqrt{2}}}$. However, they mentioned that

“The first variant is more natural because $\phi(N) = p^{r-1}(p-1)(q-1)$, where $\phi$ is Euler’s totient function.”

For the first variant, Takagi in [21] proved that when $d \leq N^{\frac{1}{\sqrt{3}+\sqrt{2}}}$, one can factor $N$. Latter in PKC 2004, May [16] improved this bound. He proved that if decryption exponent $d < N^{\frac{1}{3\sqrt{2}}} \land d < N^{\frac{1}{r-1}}$, the factorization can be done in polynomial time.

In this paper we consider $r = 2$ only as moduli of the form $N = p^2q$ are used frequently in Cryptography and therefore they are of practical interest. When $r = 2$, bound of $d$ in [16] is

$$N^{\text{max}(\frac{1}{2}, \frac{1}{\sqrt{2}})} = N^{\frac{1}{2}} \approx N^{0.22}.$$  

In this paper we have improved this bound. We consider the polynomial

$$f_r(x, y, z) = 1 + x(N - y^2 - yz + y)$$

over $\mathbb{Z}_r$ whose root is $(x_0, y_0, z_0) = (k, p, q)$, where $ed = 1 + k\phi(N)$. We have showed that one can factor $N$ in polynomial time as long as $d < N^{0.395}$. Note that $\sqrt{\frac{2}{3}}z_0 = N$. We use this fact towards our improvement. Durfee and Nguyen [7] used this type of approach to break RSA variants proposed in [20].

The organization of the paper is as follows. In Section 2, we present brief preliminaries. In Section 3, we present our result on multi prime RSA. We present our experimental results in Section 4. Section 5 concludes the paper.

2 Preliminaries

Let us present some basics on lattice reduction techniques. Consider the linearly independent vectors $u_1, \ldots, u_w \in \mathbb{Z}^n$, when $w \leq n$. A lattice, spanned by $\{u_1, \ldots, u_w\}$, is the set of all integer linear combinations of $u_1, \ldots, u_w$. The dimension of the lattice is $w$. A lattice is called full rank when $w = n$. Let $L$ be a lattice spanned by linearly independent vectors $u_1, \ldots, u_w$, where $u_1, \ldots, u_w \in \mathbb{Z}^n$. By $u'_1, \ldots, u'_w$, we denote the vectors obtained by applying the Gram-Schmidt process to the vectors $u_1, \ldots, u_w$.

The determinant of $L$ is defined as $\det(L) = \prod_{i=1}^w \|u'_i\|$, where $\|\cdot\|$ denotes the Euclidean norm on vectors. Given a polynomial $g(x, y) = \sum a_{ij} x^i y^j$, we define the Euclidean norm as $\|g(x, y)\| = \sqrt{\sum a_{ij}^2}$ and infinity norm as $\|g(x, y)\|_{\infty} = \max_{i,j} |a_{ij}|$.

It is well known that given a basis $u_1, \ldots, u_w$ of a lattice $L$, LLL algorithm [14] can find a new basis $b_1, \ldots, b_w$ of $L$ with

$$\|b_1\| \leq \|b_2\| \leq \cdots \leq \|b_i\| \leq \sqrt{\frac{\log(w-1)}{w}} \det(L)^{\frac{1}{w+1}}, \quad \text{for } i = 1, \ldots, w.$$  

Run time LLL is polynomial in the lattice dimension $w$ and the bitsize of the entries of the matrix $M$ corresponding to $L$.

In [5], Coppersmith discussed new techniques to find small integer roots of polynomials in a single variable modulo $n$, and of polynomials in two variables over the integers. The idea of [5] extends to more than two variables also, but the method becomes heuristic. The following theorem due to Howgrave-Graham [10] is also relevant to the idea of [5].
Theorem 1 Let \( g(x,y,z) \) be a polynomial which is a sum of \( \omega \) many monomials. Suppose \( g(x_0,y_0,z_0) \equiv 0 \mod n \), where \( |x_0| < X \), \( |y_0| < Y \) and \( |z_0| < Z \). If \( \| g(xY, yZ) \| < \frac{n}{\sqrt{m}} \), then \( g(x_0,y_0,z_0) = 0 \) holds over integers.

We apply Gröbner Basis based technique to solve the roots of polynomials. Though our technique works in practice as noted from the experiments we perform, theoretically this may not always happen. Thus we formally state the following heuristic assumption, that we will require for our theoretical results.

Assumption 1 Take integers \( i,n \) such that \( i \geq n \). Consider a set of \( i \) many polynomials \( \{ f_1, f_2, \ldots , f_i \} \) on \( n \) variables having the root \( (x_1,x_2,\ldots,x_n) \). Let \( J \) be the ideal generated by \( \{ f_1, f_2, \ldots , f_i \} \). Then we will be able to collect the root efficiently from the Gröbner Basis of \( J \).

Note that the time complexity of the Gröbner Basis computation is in general double-exponential in the degree of the polynomials [6]. Based on the background discussed above, let us move to our technical results.

3 Small \( d \) Attack on Prime Power RSA

Theorem 2 Let \( N = p^2q \) be an RSA modulus. Let the public exponent \( e \) and private exponent \( d \) satisfies \( ed \equiv 1 \mod \phi(N) \). Then under Assumption 1, \( N \) can be factored in polynomial time if \( d \leq N^{0.395} \).

Proof We have \( ed \equiv 1 \mod \phi(N) \) where \( N = p^2q \). So we can write \( ed = 1 + k(N - p^2 - pq + p) \). Now we want to find the root \( (x_0,y_0,z_0) = (k,p,q) \) of the polynomial

\[
f(x,y,z) = 1 + x(N - p^2 - yz + y).
\]

Let \( d \approx N^\delta \) and \( e \) is of order \( N \). Then \( k \approx N^\delta \). Since \( p,q \) are of same bit size, we have \( p \approx q \approx N^t \). Let \( X = N^\delta, Y = Z = N^t \). Clearly, \( (X,Y,Z) \) provides the upper bounds of the elements in the root \( (x_0,y_0,z_0) \), neglecting any small constant. Note that \( y_0^2z_0 = N \). We use this fact during our lattice construction.

For integers \( m,a,t \geq 0 \), we define following polynomials

\[
g_{i,j,k}(x,y,z) = x^i y^k z^j f_i(x,y,z)
\]

where \( i = 0, \ldots , m \), \( j = 1, \ldots , m - i \), \( k = j + 1, j + 2 \) and

\[
g_{i,0,k}(x,y,z) = y^k f_i(x,y,z)
\]

where \( i = 0, \ldots , m \), \( k = 0, \ldots , t \).

Now each occurrence of the monomial \( y^2z \) in \( g_{i,j,k} \) should be replaced by \( N \). After this process, let the new polynomial be \( h_{i,j,k} \). Suppose the coefficient of the monomial \( x^i y^{k+2j-2l} z^{j+a-i} \) in \( h_{i,j,k} \) be \( A \), where \( l = \min \left\{ \left\lfloor \frac{k+2j}{2} \right\rfloor, j + a \right\} \).

Let \( AB \equiv 1 \mod e^m \). Now define set of polynomials

\[
h_{i,j,k}(x,y,z) = BH_{i,j,k}(x,y,z)e^{m-j}.
\]

Note that \( h_{i,j,k}(x_0,y_0,z_0) \equiv 0 \mod e^m \).

Next, we form a lattice \( L \) by taking the coefficient vectors of the shift polynomials \( h_{i,j,k}(xX, yY, zZ) \) as basis.
Now dimension \( w \) is \( L \) will be

\[
w = \sum_{i=0}^{m} \sum_{j=1}^{m-i} \sum_{l=0}^{j} 1 + \sum_{i=0}^{m} \sum_{l=0}^{1} 1 = \frac{3}{2}m^2 + mt + o(m).
\]

There is an ordering on polynomials \( h_{i,j,k}(x,y,z) \) such that each new polynomial introduces exactly one new monomial \( s^{i+j}l^{k+2i-2j-z^i-y^j}z^{i+j-1} \), where \( i = \min \left\{ \left\lfloor \frac{k+2i}{2} \right\rfloor, j + a \right\} \). So, the determinant of \( L \) be

\[
\det(L) = \left( \prod_{i=0}^{m-j} \left( \prod_{j=1}^{m-i} x^{i+j}y^{k+2i-2j}z^{j+w-h} \right) \right) \times \left( \prod_{i=0}^{l} \left( \prod_{k=0}^{m-l} X^{i+y^{k+2i-2j}Z^{i+l}} \right) \right),
\]

where \( l_1 = \min \left\{ \left\lfloor \frac{k+2i}{2} \right\rfloor, j + a \right\} \) and \( l_2 = \min \left\{ \left\lfloor \frac{k+2i}{2} \right\rfloor, a \right\} \).

Let the determinant of \( L \) be \( \det(L) = X^{i+y^{k+2i}Z^{i+l}} \).

Now

\[
s_x = \sum_{i=1}^{m} \sum_{j=1}^{m-i} 3(i+j) + \sum_{i=0}^{m} \sum_{l=0}^{1} i = m^3 + \frac{m^3}{2} + o(m^3).
\]

Similarly,

\[
s_x = \sum_{i=1}^{m} \sum_{j=1}^{m-i} 3(m-i) + \sum_{i=0}^{m} \sum_{l=0}^{1} (m-i) = m^3 + \frac{m^3}{2} + o(m^3).
\]

During the calculations of \( s_x, s_z \), we assume \( \frac{1}{2} \leq a \leq m \).

We have,

\[
s_y = \sum_{i=0}^{m} \sum_{j=1}^{m-i} 3(i+j) \left( k + 2i - 2 \min \left( \left\lfloor \frac{k+2i}{2} \right\rfloor, j + a \right) \right)
\]

\[
+ \sum_{i=0}^{m} \sum_{l=0}^{1} \left( k + 2i - 2 \min \left( \left\lfloor \frac{k+2i}{2} \right\rfloor, a \right) \right)
\]

\[
= s^{(1)}_x + s^{(2)}_z, \text{ say.}
\]

Now,

\[
s^{(1)}_x \approx \sum_{i=0}^{m} \sum_{j=1}^{m-i} \sum_{l=0}^{j} \left( k + 2i - 2j - 2a \right)
\]

\[
\approx \sum_{i=0}^{m} \sum_{j=1}^{2i-2a} \sum_{l=0}^{j} \left( k + 2i - 2j - 2a \right) + \sum_{i=0}^{m} \sum_{j=1}^{m-i} \sum_{l=0}^{j} \left( k + 2i - 2j - 2a \right)
\]

\[
= \frac{2m^3}{3} - 2m^2a + 2ma^2 - \frac{2a^3}{3} + o(m^3).
\]
Again,

\[ s_j^{(2)} \approx \sum_{i=0}^{m} \sum_{k=\lfloor (0,2a-2i) \rfloor}^{k+2i-2a} \left( k + 2i - 2a \right) \]
\[ = \sum_{i=0}^{m} \sum_{k=\lfloor \frac{k}{2} \rfloor}^{k+2i-2a} \left( k + 2i - 2a \right) + \sum_{i=0}^{m} \sum_{k=0}^{k+2i-2a} \left( k + 2i - 2a \right) \]
\[ = m^2 t - 2amt + a^2 t + \frac{mt^2}{2} + \frac{at^2}{2} + \frac{t^3}{12} + o(m^3). \]

Hence we have

\[ s_j = \frac{2m^3}{3} - 2m^2 a + 2ma^2 - \frac{2a^3}{3} + m^2 t - 2mat + a^2 t + \frac{mt^2}{2} - \frac{at^2}{2} + \frac{t^3}{12} + o(m^3). \]

Now we will calculate \( s_{2j} \).

We have

\[ s_{2j} = \sum_{i=0}^{m} \sum_{j=0}^{j+2i} \sum_{k=\lfloor \frac{k}{2} \rfloor}^{k+2i-2j} \left( j + a - \frac{k+2i}{2} \right) + \sum_{i=0}^{m} \sum_{j=0}^{j+2i} \sum_{k=\lfloor \frac{k}{2} \rfloor}^{k+2i-2j} \left( a - \frac{k+2i}{2} \right) \]
\[ = s_{2j}^{(1)} + s_{2j}^{(2)}, \text{ say.} \]

Now

\[ s_{2j}^{(1)} \approx \sum_{i=0}^{m} \sum_{j=0}^{j+2i} \sum_{k=\lfloor \frac{k}{2} \rfloor}^{k+2i-2j} \left( j + a - \frac{k+2i}{2} \right) + \sum_{i=0}^{m} \sum_{j=0}^{j+2i} \sum_{k=\lfloor \frac{k}{2} \rfloor}^{k+2i-2j} \left( a - \frac{k+2i}{2} \right) \]
\[ = \frac{m^3}{12} + \frac{m^2 a}{2} + ma^2 - \frac{a^3}{3} + o(m^3). \]

Again

\[ s_{2j}^{(2)} \approx \sum_{i=0}^{m} \sum_{k=0}^{k+2i} \left( a - \frac{k+2i}{2} \right) + \sum_{i=0}^{\lfloor \frac{2a-2i}{2} \rfloor} \sum_{k=0}^{k+2i} \left( a - \frac{k+2i}{2} \right) \]
\[ = \frac{a^2 t}{2} - \frac{at^2}{4} + \frac{t^3}{24} + o(m^3). \]

Hence,

\[ s_j = \frac{m^3}{12} + \frac{m^2 a}{2} + ma^2 - \frac{a^3}{3} + \frac{a^2 t}{2} - \frac{at^2}{4} + \frac{t^3}{24} + o(m^3). \]

One gets \((x_0, y_0, z_0)\) (under Assumption 1 and following Theorem 1) using lattice reduction over \( L \), if \( \det(L)^{\frac{1}{2}} < e^m \), i.e., when \( \det(L) < e^m \) (neglecting the lower order terms).
Let \( a = \tau_1 m \) and \( t = \tau_2 m \), where \( \tau_1, \tau_2 \) are non-negative real numbers. Now putting the values of \( \det(L) \) and \( w \) in the condition \( \det(L) < e^{mw} \), we need

\[
\eta(\tau_1, \tau_2) = \frac{\tau_1^3}{3} + \frac{\tau_1^2}{2} + \frac{\tau_1 \tau_2^2}{4} + \frac{\tau_2^3}{24} + \frac{\tau_1}{3} - \frac{\tau_1^2}{6} + \frac{\tau_2}{2} \delta
\]

For a fixed \( \delta \), we will take the partial derivative of \( \eta \) with respect to \( \tau_1, \tau_2 \) and equate each of them to 0, we get \( \tau_1 = \frac{3}{4} - \frac{\delta}{\tau_1^2} \) and \( \tau_2 = -3 \delta + \frac{\sqrt{7-9\delta}}{\tau_2} + \frac{1}{\tau_2^3} \). Putting these values of \( \tau_1, \tau_2 \) in \( \eta \), we get \( \eta < 0 \) when \( \delta \leq 0.395 \). That is when \( \delta = 0.395 \), \( \tau_1 \) will be 0.657 and \( \tau_2 \) will be 1.211 with \( \eta(\tau_1, \tau_2) < 0 \). Also for these values of \( \tau_1 \) and \( \tau_2 \), \( \frac{1}{\tau_2} \leq a \leq \frac{1}{\tau_1} \). Hence the result.

Let us illustrate the lattice generation technique for \( m = 1, a = 1, t = 1 \). We use the polynomials \( xyz^2 e, xy^2 z^2 e, xy^3 z^2 e, ze, yze, z f e, yzf e \) and build the following lattice \( L \), as discussed before. In this case, the lattice dimension turns to be 7. The ‘−’ marked places contain non-zero elements, but we do not write those as those elements do not contribute in the calculation of the determinant.

<table>
<thead>
<tr>
<th>poly</th>
<th>( xyz^2 e )</th>
<th>( xz )</th>
<th>( xyz )</th>
<th>( z )</th>
<th>( ye )</th>
<th>( x )</th>
<th>( xy )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( xy^2 z^2 e )</td>
<td>( XY^2 e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( xy^3 z^2 e )</td>
<td>( XYZ e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( ze )</td>
<td>( Z e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( ye )</td>
<td>( Y e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( zf e )</td>
<td>( Yf e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( yzf e )</td>
<td>( Yfz e )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 3.0.1 Comparison with the Existing work: |

When \( r = 2 \), \( \max \left\{ \frac{r}{(r+1)^2}, \left(\frac{r}{r+1}\right)^2 \right\} = \frac{r}{(r+1)^2} = 0.22 \). Hence as long as \( d < N^{0.22} \), one can factor \( N \) in polynomial time using the idea of [16]. In Theorem 2, we have improved this bound upto 0.395. However, as we work with low lattice dimensions, the theoretical bounds of \( d \) presented in Theorem 2 may not be reached. In Table 1, we present few numerical values of \( \delta \) for different values of \( m, a, t \).

4 Experimental Results

We have implemented the code in SAGE 4.1 on a Linux Ubuntu 8.10, Dual CORE Intel(R) Pentium(R) D CPU 1.83 GHz, 2 GB RAM, 2 MB Cache machine. Let us first present one example to explain our improvements.

Example 1 We consider 1000 bits \( N \), where \( p, q \) are as follows:

\begin{tabular}{l}
1991210136082632006191348942343145617245670253210185535341816
585775624789103417058574282566895119, and
215041477465507573649063838184185137844906941923771876800489214
91313532676580654675656899878204179.
\end{tabular}
Small Secret Exponent Attack on RSA Variant with Modulus \(N = p^2q\)

<table>
<thead>
<tr>
<th>(m)</th>
<th>(a)</th>
<th>(t)</th>
<th>(\delta)</th>
<th>Lattice Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
<td>0.25</td>
<td>87</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>0.27</td>
<td>119</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>0.30</td>
<td>198</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>14</td>
<td>0.32</td>
<td>330</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>13</td>
<td>0.33</td>
<td>416</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>16</td>
<td>0.34</td>
<td>570</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>21</td>
<td>0.35</td>
<td>855</td>
</tr>
<tr>
<td>22</td>
<td>14</td>
<td>26</td>
<td>0.36</td>
<td>1380</td>
</tr>
<tr>
<td>31</td>
<td>20</td>
<td>57</td>
<td>0.37</td>
<td>2704</td>
</tr>
<tr>
<td>52</td>
<td>33</td>
<td>60</td>
<td>0.38</td>
<td>7367</td>
</tr>
</tbody>
</table>

Table 1 Numerical values of \(\delta\) for different values

The public encryption exponent \(e\) and the private decryption exponent \(d(\approx N^{0.272})\) are as follows:

214631002612569534998147253816228009577741660929349591512096112
2672283086890719496926112349317794553765749071947778102754267683
7971752090251541598086981607859943819842383752527891575532973
00959618529353416642132983006035285893005093784981129594123993218
28649754529923471850351990939915691895874849, and
455720369281483631593617574727820599342387644916431063628488034
709144741946250173.

We work with \(m = 6, a = 4, t = 7\). So lattice dimension will be \(w = 119\). Lattice reduction takes 23766.87 seconds.

Now we present more experimental results in tabular form in Table 2. In all the cases, we considered \(\delta > 0.22\) as the bound of [16] is \(\delta < 0.22\). We could successfully collect the roots in all the experiments.

<table>
<thead>
<tr>
<th>(m)</th>
<th>(a)</th>
<th>(t)</th>
<th>(\delta)</th>
<th>Lattice Dimension</th>
<th>LLL time (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0.230</td>
<td>69</td>
<td>1569.22</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>0.246</td>
<td>81</td>
<td>2751.14</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
<td>0.250</td>
<td>87</td>
<td>3664.03</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>0.265</td>
<td>105</td>
<td>12504.29</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>0.272</td>
<td>119</td>
<td>23766.87</td>
</tr>
</tbody>
</table>

Table 2 Experimental Results for 1000-bit \(N\).

5 Conclusion

In this paper, we have considered the RSA modulus \(N = p^2q\). We have proved that if the decryption exponent \(d \leq N^{0.395}\), one can factor \(N\) in polynomial time. This improves the existing bound \(N^{0.22}\). In the full version of the paper, we will consider the general case \(N = p^r q\) where \(r \geq 2\).
References

Abstract  We provide the differential spectra of differentially 6-uniform functions among the family of power functions $x \mapsto x^{2^t-1}$ defined in $\mathbb{F}_{2^n}$. We show that the functions $x \mapsto x^{2^t-1}$ when $t = \frac{n-1}{3}$, $\frac{n+3}{3}$ with odd $n$ and when $t = \frac{kn+1}{3}$, $\frac{(3-k)n+2}{3}$ with $kn \equiv 2 \mod 3$ have differential spectra similar to the one of the function $x \mapsto x^7$ which belongs to the same family. To provide the differential spectra for these functions, a recent result of Helleseth and Kholosha regarding the number of roots of polynomials of the form $x^{2^t+1} + x + a$ is used. A discussion regarding the non-linearity and the algebraic degree of this family of exponents is provided.

Keywords  Differential uniformity · Differential spectrum · Kloosterman sum · Power function · Roots of trinomial · $x \mapsto x^{2^t-1}$ · Dickson polynomial.

1 Introduction

Differential cryptanalysis is the first statistical attack proposed for breaking iterated block ciphers. Its publication [2] then gave rise to numerous works which investigated the security offered by different types of functions regarding differential attacks. This security is quantified by the so-called differential uniformity of the Substitution box used in the cipher [23]. Most notably, finding appropriate S-boxes which guarantee that the cipher using them resists differential attacks has been a major topic for the last twenty years.

Power functions, i.e., monomial functions, form a class of suitable candidates since they usually have a lower implementation cost in hardware. Also, their particular algebraic structure makes the determination of their differential properties easier. However, there are only few power functions with proved low differential
uniformity. Up to equivalence, there are two large families of such functions: a subclass of the quadratic power functions (a.k.a. Gold functions) and a subclass of the so-called Kasami functions. Both of these families contain some permutations which are APN over $\mathbb{F}_{2^n}$ for odd $n$ and differentially 4-uniform for even $n$. The other known power functions with a low differential uniformity correspond to “sporadic” cases in the sense that the corresponding exponents vary with $n$ [18] and they do not belong to a large class: they correspond to the exponents defined by Welch [13,7], by Niho [12,19], by Dobbertin [14], by Bracken and Leander [6], and to the inverse function [22].

If the conjecture of [4] concerning the list of monomials differentially 4-uniform is true, we can consider that the differential spectrum of all power functions differentially 4-uniform have been provided in the past. For some cryptographic applications, using a differentially 6-uniform power function instead of a differentially 2- or 4-uniform does not drastically reduce the security of the cipher. Power functions differentially 6-uniform can then present some cryptographic interest in particular if non-linear and algebraic properties are also relatively good for cryptographic applications.

Simulations of [5] show that, for $17 \leq n \leq 31$, all power functions differentially 6-uniform belong to the family $G_t(x) = x^{2^t-1}$. In [5], properties of the differential spectra of such functions were investigated. In particular, it was shown that the differential spectrum of the power function $G_t(x) = x^{2^t-1}$ and the one of the power function $G_s(x)$ with $s = n - t + 1$ are related to each other. In the same paper, the differential spectrum of the power functions $x \mapsto x^7$ and $x \mapsto x^{2n-2}-1$ on the field $\mathbb{F}_{2^n}$ is extracted.

In this paper, we give explicit formulas for the differential spectrum of $G_t$ when $n$ is odd and $t = \frac{n-1}{2}$, $\frac{n+3}{4}$, or when $kn \equiv 2 \mod 3$ ($k = 1$ or $k = 2$) and $t = \frac{kn+1}{4}$, $\frac{(2-k)n+2}{4}$. All of these functions are differentially 6-or 8-uniform. We show in particular that their differential spectrum, which can be expressed in terms of the Kloostermann sum, is similar to the one of the function $G_3(x) = x^7$. While computing the differential spectrum of the function $x^7$, an important result of [1] regarding the number of roots of the polynomial $x^3 + x + a$ was used. In this paper, as the degree of the derivative depends on $t$, we use a generalisation of this result published recently by Helleseth and Kholosha [16,17]. Differential spectrum provided in Theorem 3 and 5 are proofs of Conjecture 8.9 and 8.10 proposed in [3]. A relation between derivative of monomials and reversed Dickson polynomials is also mentioned in this paper.

The following of this paper is organised as follows. In Section 2, definitions and results relevant to this work are recalled. While in Section 3, the functions $G_t$ with $t = \frac{kn+1}{4}$ and $t = \frac{(2-k)n+2}{4}$ are studied, Section 4 presents the differential spectra of the functions $G_t$ with $t = \frac{n-1}{2}$ and $\frac{n+3}{4}$. In Section 5, a discussion regarding the inverse of these functions $G_t$, their algebraic degree and their non-linearity is given. Section 6 concludes this paper.
2 Preliminary

2.1 Functions over $\mathbb{F}_{2^n}$ and their derivatives

Any function $F$ from $\mathbb{F}_{2^n}$ into $\mathbb{F}_{2^n}$ can be expressed uniquely as a univariate polynomial in $\mathbb{F}_{2^n}[x]$ of degree at most $2^n - 1$. The algebraic degree of $F$, denoted by $\deg(F)$, is the maximal Hamming weight of the 2-ary expansions of its exponents. In this paper, we identify a polynomial of $\mathbb{F}_{2^n}[x]$ with the corresponding function over $\mathbb{F}_{2^n}$.

In the following, we denote by $\text{Tr}$ the absolute trace on $\mathbb{F}_{2^n}$, i.e.,

$$\text{Tr}(\beta) = \beta + \beta^2 + \cdots + \beta^{2^n-1}, \quad \beta \in \mathbb{F}_{2^n}.$$ 

In the whole paper, $\#E$ is the cardinality of any set $E$. To simplify the notation, we also denote by $\mathcal{F}$ the set $\mathbb{F}_{2^n}\setminus\{0,1\}$.

The resistance to differential cryptanalysis is related to the following quantities, introduced by Nyberg and Knudsen [23,22].

**Definition 1** Let $F$ be a function from $\mathbb{F}_{2^n}$ into $\mathbb{F}_{2^n}$. For any $a \in \mathbb{F}_{2^n}$, the derivative of $F$ with respect to $a$ is the function $D_a F$ from $\mathbb{F}_{2^n}$ into $\mathbb{F}_{2^n}$ defined by

$$D_a F(x) = F(x + a) + F(x), \quad \forall x \in \mathbb{F}_{2^n}.$$ 

The resistance to differential cryptanalysis is related to the following quantities.

The resistance of a cipher to differential attacks and to its variants is quantified by some properties of the derivatives of its S-boxes (substitution)-box, in the sense of the following definition.

**Definition 2** Let $F$ be a function from $\mathbb{F}_{2^n}$ into $\mathbb{F}_{2^n}$. For any $a$ and $b$ in $\mathbb{F}_{2^n}$, we denote

$$\delta(a,b) = \#\{x \in \mathbb{F}_{2^n}, D_a F(x) = b\}.$$ 

The differential uniformity of $F$ is $\delta(F) = \max_{a \neq 0, b \in \mathbb{F}_{2^n}} \delta(a,b)$. Functions for which $\delta(F) = 2$ are said to be almost perfect non-linear (APN).

In this paper, we focus on the case where the S-box is a power function, i.e., a monomial function on $\mathbb{F}_{2^n}$. In other words, $F(x) = x^d$ over $\mathbb{F}_{2^n}$. In the case of such a power function, the differential properties can be analysed more easily since, for all $a \neq 0$, $\delta(a,b) = \delta(1,b/a^d)$. Then, when $F : x \mapsto x^d$ is a monomial function, the differential characteristics of $F$ are determined by the values $\delta(b) = \delta(1,b), b \in \mathbb{F}_{2^n}$.

The differential spectrum of $F$ can be defined as follows.

**Definition 3** Let $F(x) = x^d$ be a power function on $\mathbb{F}_{2^n}$. We denote by $\omega_i$ the number of output differences $b$ that occur $i$ times:

$$\omega_i = \#\{b \in \mathbb{F}_{2^n} \mid \delta(b) = i\}.$$ 

The differential spectrum of $F_d$ is the set of the $\omega_i$: $\mathcal{S} = \{\omega_0, \omega_2, \ldots, \omega_{\delta(F)}\}$.

Obviously, the differential spectrum satisfies

$$\sum_{i=0}^{\delta(F)} \omega_i = 2^n \quad \text{and} \quad \sum_{i=2}^{\delta(F)} (i \times \omega_i) = 2^n, \quad (1)$$ 

where $\omega_i = 0$ for odd $i$.

As explained in [20], derivatives of monomial $F_d : x \mapsto x^d$ are linked with the Dickson polynomial $D_d$ (see [11] for definition). We have $(x + 1)^d + x^d =
In [5], a complete definition of the differential spectrum of the function \( G \)

Theorem 5 of [5], it is shown that:

as the sequence of values \( \lambda \) determined only by \( \omega \) is provided. As this function is differentially 6-uniform, and the value of \( \tau \) is invertible, we denote by \( F \) defined in the field \( \mathbb{F} \) are often particular cases. In the following we say that a function \( D \) over \( \mathbb{F} \) are co-prime, the function \( G \) function

In [5], different properties of the differential spectra of the functions \( G_t(x) = x^{2^t-1} \), defined in the field \( \mathbb{F}_{2^n} \), were investigated. In particular it was shown that the quantities \( \delta(b) \) can be computed as follows:

\[
\delta(0) = 2^{s\gcd(t,n)} - 2, \quad \delta(1) = 2^{s\gcd(t-1,n)}, \quad \forall b \neq 0, 1 \quad \delta(b) = N_b - 2,
\]

were \( N_b \) is the number of roots of the linear polynomial \( P_b(x) = x^{2^t} + bx^2 + (b+1)x \) over \( \mathbb{F}_{2^n} \). The problem of determining the restricted differential spectrum of the function \( G_t \) is then equivalent to the problem of finding the number of roots of the linear polynomial \( P_b(x) \) for \( b \in \mathbb{F} \). In particular \( \omega'_i \neq 0 \) if and only if \( i = 2^t - 2 \) for some \( r \). As 0 and 1 are simple roots of this polynomial, by setting \( y = x^2 + x \), we obtain that \( N_0, b \in \mathbb{F} \), is equal to twice the number of roots of the following system (see Theorem 3 of [5]) in \( \mathbb{F}^{2_n}_2 \):

\[
E_b : \quad \begin{cases} Q(y) = by, & Q(y) = \sum_{i=0}^{t-1} y^{2^i} \end{cases}
\]

In [15], the reversed Dickson polynomials when \( d = 2^t - 1 \) are studied. Assuming \( y \neq 0 \), we notice that \( RD_{2^t-1}(y) = Q(y)/y \).

In [5], it is shown that the restricted differential spectrum of the functions \( G_t \) and \( G_s \) with \( s = n - t + 1 \) are equal. Only the values \( \delta(0) \) and \( \delta(1) \) can differ and are defined by (2). The function \( G_t \) is called the symmetric of \( G_t \).

When \( n \) and \( t \) are co-prime, the function \( G_t \) is a permutation. In Theorem 7 of [21], the inverse of the exponent \( 2^t - 1 \) is extracted. In the following, when \( G_t \) is invertible, we denote by \( \tau \) the inverse of \( 2^t - 1 \) mod \( 2^n - 1 \):

\[
\tau = \sum_{i=0}^{t^{-1}-1} \text{mod } n 2^{i} \text{ mod } 2^n - 1.
\]

In [5], a complete definition of the differential spectrum of the function \( G_3(x) = x^7 \) is provided. As this function is differentially 6-uniform, and the value of \( \omega_4 \) is determined only by \( \delta(1) \), the complete differential spectrum can be derived from \( \omega_0 \). Using (1), we have \( \omega_2 = 2^n - 3 \omega_0 - 2 \omega_4 \) and \( \omega_0 = 2^n + 2 \omega_0 + \omega_4 \). In Theorem 5 of [5], it is shown that:

- if \( n \) is odd, \( \omega_0 = \frac{2^{n-2} + 1}{6} - \frac{K(1)}{8}, \omega_4 = 0 \),
- if \( n \) is even, \( \omega_0 = \frac{2^{n-2} - 4}{6} + \frac{K(1)}{8}, \omega_4 = 1 \),
where $K(1)$ is the Kloosterman sum defined by

$$K(1) = \sum_{x \in F_{2^n}} (-1)^{Tr(x^{-1} + x)},$$

with the convention that $(-1)^{Tr(x^{-1})} = 1$ for $x = 0$. To express the differential spectrum of the function $x^2$, a result from [10] which gives the number of roots of the polynomial $x^3 + x + a$, was used. In this paper, results are derived using a generalisation of [16] concerning the number of roots of the polynomial $L_a(x) = x^{2^t+1} + x + a$ in $F_{2^n}$.

**Theorem 1** (Theorem 1 of [16]) Let $t$ be a positive integer such that $t \leq n$ and $\gcd(t, n) = 1$. For any $a \in F_{2^n}$, the polynomial $L_a(x) = x^{2^t+1} + x + a$ has either none, one or three roots in $F_{2^n}$. Further $L_a$ has exactly one zero in $F_{2^n}$, namely $x_0$, if and only if $\Tr\left((1 + x_0^{-1})^t\right) = 1$. Let $M_i = \#\{a \in F_{2^n} \mid L_a \text{ has } i \text{ roots}\}$.

For $n$ odd, $M_0 = \frac{2^n + 1}{3}$, $M_1 = 2^{n-1} - 1$, $M_3 = \frac{2^{n-1} - 1}{3}$.

For $n$ even, $M_0 = \frac{2^n - 1}{3}$, $M_1 = 2^{n-1}$, $M_3 = \frac{2^{n-1} - 2}{3}$.

### 3 Functions $x \mapsto x^{2^{t-1}}$ when $t = (kn + 1)/3$

In this section, we focus on field $F_{2^n}$ where $n$ is not a multiple of 3. We define $k = 1, 2$ such that $kn \equiv 2 \mod 3$. We are interested in the computation of the differential spectra of the functions $G_t$ and $G_s$ with $t$ and $s$ as follows:

$$t = \frac{kn + 1}{3}, \quad s = \frac{(3-k)n + 2}{3}.$$  

Notice that $k$ is chosen such that $t$ and $s$ are integer values. For this value of $t$, the function $G_t$ is a permutation (which is not always the case for the function $G_s$).

The inverse of $2^t - 1$ modulo $2^n - 1$, which we denote by $\tau$, can be computed easily from (4) as $\tau = 1 + 2^t + 2^{2t}$.

To derive the differential spectrum of these functions, we provide a different formulation of (3). The following result will be used when rewriting the system.

**Lemma 1** If $n \not\equiv 0 \mod 3$, then $L_1(x) = x + x^{2^t} + x^{2^{2t}}$ has only one root $x = 0$.

**Proof** This lemma is a direct consequence of Proposition 2 of [17]. This polynomial has only one root zero if and only if $L_1(z) = z^{2^t+1} + z + 1$ ($z = x^{2^{2t}-1}$) is irreducible. Using the same notation as in the original paper, we notice that if $n \not\equiv 0 \mod 3$, we have $C_n(1) = 1$ and $Z_n(1) = 1$. As $\gcd(t, n) = 1$, $L_1$ is irreducible. \hfill $\Box$

**Theorem 2** Let $\beta = 1 + b^{-1}$. For $b \neq 0$ (if $b = 0$ then $\beta = 1$), the number of solutions of $G_t(x) + G_t(x + 1) = b$ is equal to twice the number of solutions of this system:

$$\begin{cases} L_2(v) = v^{2^t+1} + v + \beta = 0, \\ \Tr(v^\tau) = 0.
\end{cases}$$

**Proof** For the sake of simplification, we present the proof for $b \neq 0$. Let $Q(y)$ be the polynomial of (3). Summing $Q(y)^{2^i}$ for $i = 0, 1, 2$ gives:

$$Q(y) + Q(y)x^t + Q(y)x^{2t} = \sum_{i=0}^{kn} y^{x^i} = k \times \Tr(y) + y^{2^kn} = y.$$

5
From Lemma 1, \( \sum_{i=0}^{n}(Q(y) + (by))^{2i} = 0 \) if and only if \( Q(y) + by = 0 \). Thus, (3) has the same number of solutions as:

\[
(1 + b)y + (by)^{2} + (by)^{2n} = 0 \quad \text{and} \quad \text{Tr}(y) = 0.
\]

Using the substitution \( z = by \), the equation \( (1 + b)y + (by)^{2} + (by)^{2n} = 0 \) of (7) is equivalent to \( L_{\beta}(z) = 0 \) where \( \beta = 1 + b^{-1} \) and \( L_{\beta}(z) = z^{2^{n-1}} + z^{2^i} + \beta z \). The linear polynomial \( L_{\beta} \) can be decomposed as

\[
L_{\beta}(z) = z \cdot (z^{(2^{n-1})(2^i+1)} + z^{2i-1} + \beta) = z \cdot L_{\beta}(z^{2i-1}),
\]

where \( L_{\beta}(x) = x^{2^i+1} + x + \beta \). Thus, \( L_{\beta}(z) = 0 \) and \( z \neq 0,1 \) is equivalent to \( L_{\beta}(z^{2i-1}) = 0 \). Furthermore, if \( L_{\beta}(z) = 0 \), we have \( z^{2^i} + z^{2i-1} + z + y = 0 \). Thus, \( \text{Tr}(z) = \text{Tr}(y) \) so the trace condition of (7) is equivalent to \( \text{Tr}(z) = 0 \). If we let \( v = z^{2i-1} \), as we remove the trivial solution \( y = 0 \), (7) has exactly one more root than (6) meaning the same number of solutions as \( G_{t}(x) + G_{t}(x+1) + b = 0 \). □

Using this theorem, we derive the differential spectrum of the function \( G_{t} \).

**Theorem 3** Let \( G_{t} \in \mathbb{F}_{2^n}[x] \), with \( t = \frac{kn+1}{2} \) and \( k = 1 \) or 2 depending on \( n \). The function \( G_{t} \) is differentially 6-uniform. Let \( K(1) \) as defined in (5), its differential spectrum \{\( \omega_{0}, \omega_{2}, \omega_{4}, \omega_{6} \} \) is determined as follows:

- if \( n \equiv \pm 1 \mod 6 \), \( \omega_{6} = \frac{2^{n-2}+1}{6} - \frac{K(1)}{8}, \omega_{4} = 0 \).
- if \( n \equiv \pm 2 \mod 6 \), \( \omega_{6} = \frac{2^{n-2}-4}{6} + \frac{K(1)}{8}, \omega_{4} = 1 \).

In both cases, \( \omega_{2} = 2^{n-1} - 3\omega_{6} - 2\omega_{4} \) and \( \omega_{0} = 2^{n-1} + 2\omega_{6} + \omega_{4} \).

**Proof** We first compute the value of \( \delta(0) \) and \( \delta(1) \) separately. As \( G_{t} \) is a permutation we have \( \delta(0) = 0 \). The value of \( \delta(1) \) depends on the value of \( n \) modulo 6: if \( n \equiv \pm 2 \mod 6 \), \( \delta(1) = 4 \) and \( \omega_{4} = 1 \), if \( n \equiv \pm 1 \mod 6 \), \( \delta(1) = 2 \) and \( \omega_{4} = 0 \).

According to Theorem 2, the differential spectrum of \( G_{t} \) can be computed by studying the number of roots of (6). According to Theorem 1, as \( \gcd(t,n) = 1 \), \( L_{\beta} \) can only have 0, 1 or 3 roots; meaning that for \( b \neq 1 \), \( \delta(b) = 0,2,6 \). By determining the number of \( \beta \in \mathbb{F}_{2^n} \) \( (b \neq 1) \) such that (6) does not have any roots, we can derive the complete differential spectrum (see. (1)). The number of \( b \) such that \( G_{t}(x) + G_{t}(x+1) = b \) is irreducible is equal to the number of \( \beta \) such that the system has no solution and is denoted by \( \omega_{0} \). We notice that:

- If \( L_{\beta} \) has three roots, \( v_{1}, v_{2} \) and \( v_{3} \), since \( v_{1}^{i}, v_{2}^{i}, v_{3}^{i} \) are roots of \( L_{\beta} \), we have \( v_{1}^{i} + v_{2}^{i} + v_{3}^{i} = 0 \) and at least one of them is such that \( \text{Tr}(v_{1}^{i}) = 0 \). Therefore if \( L_{\beta} \) has three roots, (6) has a least one solution.
- According to Theorem 1, \( v_{0} \) is the unique root of some \( L_{\beta} \) if and only if \( \text{Tr}(1+v_{0}^{-1}) = 1 \).

From these observations, we deduce that (6) has no solution if and only if \( L_{\beta} \) is irreducible or if \( L_{\beta} \) has one root \( v \) such that \( \text{Tr}(v') \neq 0 \) and \( \text{Tr}(1+v_{0}^{-1}) = 1 \). As \( \omega_{0} \) corresponds to the number of \( \beta \) such that the system has no roots, we have:

\[
\omega_{0} = M_{0} + \#\{v \in \mathbb{F}, \text{Tr}(v') = 1, \text{Tr}(1+v^{-1}) = 1\}.
\]

Since \( x \mapsto x' \) is a permutation, using a similar method to [5] by separating with regards to the parity of \( n \), we obtain that the set on the right is of size
2^n-2 + (-1)^n K(1)/4. This allows us to conclude that
\[ \omega_0 = \frac{2^n + (-1)^{n+1}}{3} + 2^n - 2 + (-1)^n \frac{K(1)}{4}. \]

The complete differential spectrum can be computed using (1).

Using the differential spectrum of the function \( G_t \), the differential spectrum of the symmetric function \( G_s \), \( s = \frac{\beta - k}{2} \) can be derived easily (Theorem 4 of [5]). Notice that in that case \( G_s \) is a permutation if and only if \( n \equiv \pm 1 \mod 6 \) \( (\delta(0) = 0) \). If \( n \equiv \pm 2 \mod 6 \), we have \( \delta(0) = 2 \). In all cases we have \( \delta(1) = 2 \) and the function \( G_s \) is differentially 6-uniform.

4 Functions \( x \mapsto x^{2^t-1} \) when \( t = (n - 1)/2 \)

In this section, we focus on field \( \mathbb{F}_{2^n} \) with odd \( n \). Theorem 9 of [5] states that the permutation \( G_t(x) = x^{2^t-1} \) with \( t = \frac{n-1}{2} \) is locally differentially 6-uniform. The symmetric function is the function \( G_s \) with \( s = \frac{\beta + 1}{2} \). In this section, we provide the complete differential spectrum of these functions. Theorem 5 shows that their differential spectrum is similar to the one of the function \( G_3(x) = x^7 \).

In comparison with the previous section, a direct reduction of the problem to a system similar to (6) does not lead easily to the derivation of the differential spectrum.

As \( \gcd(\frac{n-1}{2}, n) = 1 \), the functions \( G_t : x \mapsto x^{2^t-1} \) studied in this section are permutations. As the differential spectrum of a function is equal to the differential spectrum of its inverse, we study the differential spectrum of the function \( \Gamma_t : x \mapsto x^T \) with \( T = -2 - 2^{t+1} \). In this section, a property of the reversed Dickson polynomial when \( d = 2^{t+1} - 1 \) is used to derive the differential spectrum.

**Theorem 4** Let \( b \neq 1 \) and let \( \beta = b^{1-2^t} \). The number of roots of \( \Gamma_t(x) + \Gamma_t(x+1) = b \) is equal to twice the number of roots of:
\[
\begin{cases}
L_\beta(v) = v^{2^t+1} + v + \beta = 0,
\text{Tr}(v) = 1 + \text{Tr}(\beta).
\end{cases}
\]

**Proof** We denote by \( \delta(b) \) the number of roots of \( \Gamma_t(x) + \Gamma_t(x+1) = b \). By rewriting and simplifying the equation \( (x+1)^T + x^T = b \), we obtain that the number of roots of \( H_c(x) = x^{2^t} + x + 1 + c(x^2 + x)^{2^{t+1}} \), where \( c = b^{2^n-1} \) is \( \delta(b) + 2 \). By setting \( y = x + x^T \), this number of roots is equal to twice the number of roots of
\[
\chi_c(y) = cy^{2^t+1} + \sum_{i=0}^{t-1} y^{2^i} + 1 = 0 \quad \text{and} \quad \text{Tr}(y) = 0,
\]
which has as many solutions as:
\[
y^{-2^t} (\chi_c(y) + \chi_c(y)^{2^{t+1}}) = c^{2^t+1} y^{2^t+1} + cy + 1 = 0 \quad \text{and} \quad \text{Tr}(cy^{2^t+1}) = 1.
\]

At last, if we let \( v = yc^{2^t+1} \) and \( \beta = c^{1-2^{t+1}} \equiv b^{1-2^t} \), we obtain the result. Notice that when \( v \) is a root of \( L_\beta \), the condition \( \text{Tr}(v^{2^t+1}) = 1 \) can be written as \( \text{Tr}(v) = 1 + \text{Tr}(\beta) \).

As before, Theorem 1 gives us the number of solutions of this system if we do not take the condition over the trace into account. Before determining the differential spectrum we introduce the following lemma.
Lemma 2 Let \( A(l) = \sum_{i=1}^{t} i^{2^{i-1}} \) be a function of \( F_{2^n} \). The set of the \( x \) being the unique roots of polynomial \( \mathcal{L}_\beta \) for some \( \beta \) is equal to the image of \( \mathcal{F}_0 = \{ x \in \mathcal{F} \mid \text{Tr}(x) = 0 \} \) by the function \( l \mapsto 1/A(l) \).

Proof From [16], \( \mathcal{L}_\beta \) has 0, 1 or 3 roots. Let \( \beta \) be such that \( \mathcal{L}_\beta \) has 3 roots. We denote by \( x \) one of the roots. As \( \mathcal{L}_\beta \) has 3 roots, it exists \( \gamma \in \mathcal{F} \) such that \( y = \gamma x \) is also a root of \( \mathcal{L}_\beta \). Thus, \( x^{2^{j+1}} + x = (\gamma \cdot x)^{2^{j+1}} + \gamma \cdot x \). After simplification, we obtain that \( x^{2^j} = (1 + \gamma)/(1 + \gamma^{2^{j+1}}) \). By setting \( \gamma = \alpha^{2^{j+1}-2} \), we have \( x^{2^j} = (\alpha^2 + \alpha^{2^{j+1}})/(\alpha + \alpha^2) \). By setting \( l = \alpha + \alpha^2 \), we obtain:

\[
\mathcal{L}_\beta(x) = \mathcal{L}_\beta(y) = 0, \ (x \neq y) \iff \exists l \in \mathcal{F}_0, x^{2^j} = A(l) = \sum_{i=1}^{t} i^{2^{i-1}}.
\]

Therefore, \( \{ x \mid \mathcal{L}_\beta(x) = 0 \} \) and \( \mathcal{L}_\beta \) has 3 roots. Let \( \mathcal{L}_\beta \) be such that \( \mathcal{L}_\beta \) is irreducible or if \( \mathcal{L}_\beta \) has one root \( v \) such that \( \mathcal{L}_\beta(v^{2^{j+1}}) \neq 1 \). Hence, we have the following expression:

\[
\omega_0 = M_0 + \#(v \in \mathcal{F}, \exists l \in \mathcal{F}, v = 1/A(l), \text{Tr}(l) = 0, \text{Tr}((l \cdot A(l))^{-1})) = 1.
\]

If \( \text{Tr}(l) = 0 \) then \( \text{Tr}(l \cdot A(l)) = 0 \) and \( \lambda : l \mapsto l \cdot A(l) \) is a permutation of \( \mathcal{F}_0 \) so by setting \( l^\prime = \lambda(l) \), we have

\[
\omega_0 = M_0 + \#(l^\prime \in \mathcal{F}, \text{Tr}(l^\prime) = 0, \text{Tr}(l^\prime^{-1}) = 1).
\]
We know from [5] that the size of this set for odd \( n \) is \( 2^{n-2} - K(1)/4 \). Thus, we know that \( \omega_0 \) is such that:

\[
\omega_0 = \frac{2^n + 1}{3} + 2^{n-2} - \frac{K(1)}{4}.
\]

The complete differential spectrum can be computed using (1).

Using the differential spectrum of the function \( G_t \), the differential spectrum of the symmetric function \( G_s \) with \( s = \frac{n+3}{2} \) can be derived easily (Theorem 4 of [5]). Notice that \( G_s \) is a permutation if and only if \( n \not\equiv 0 \mod 3 \) (\( \delta(0) = 0 \)). If \( n \equiv 0 \mod 3 \), we have \( \delta(0) = 2 \). In all cases \( \delta(1) = 2 \). We conclude that the function \( G_s \) is differentially 6-uniform.

5 Overview on the function \( x \mapsto x^{2^t-1} \)

Among monomials with exponents \( 2^t - 1 \) \((t = 2, \ldots, n-1)\), several have good differential properties, as for instance the Gold function \( G_2(x) = x^3 \) and the inverse function \( G_{n-2}(x) = x^{2^{n-2}-1} \). In [5] it is conjectured that only few \( G_t \) are APN (equivalent to known APN). When \( n \) is odd, it has been proved recently in [15] that this conjecture is true and only \( G_2(x) = x^3 \) is APN. Proving the case \( n \) even of Conjecture 1 of [5] is still an open problem. In [5] and in this article, the differential uniformity and the differential spectrum for different values of \( t \) have been extracted. These functions, resumed in Table 1, are locally differentially 2-or 6-uniform meaning that for \( b \neq 0,1 \), \( \delta(b) \leq 6 \).

In Table 1, we give the list of functions \( G_t \) proved locally differentially 2-or 6-uniform. As the algebraic degree is also a criteria when designing a block cipher, in the same table, we resume the algebraic degree of the functions \( G_t \) their symmetric \( G_s \) and, when they exist, their inverses. When the function is not invertible the degree is denoted by \( \delta^* \). The differential uniformity and the permutation property depend of the value of \( n \) modulo 2, 3, 6 or 9. Restricted differential uniformity is denoted by \( \Delta = \max_{b \neq 0,1} \delta(b) \). Algebraic degree of the inverse of the function \( G_t \) has been computed using (4).

As actual block ciphers are designed using many iterations of the non-linear layer, using a function differentially 6-uniform instead of a differentially 2-or 4-uniform does not influence directly the security of the cipher in regards to differential cryptanalysis. Nevertheless, degree of the function or its inverse is important to insure some security against algebraic attacks. For the locally differentially 6-uniform ones, we notice that the algebraic degrees are related to each other. For instance, in some cases, we have \( \deg(G_{(n-1)/2}) = \deg(G_{(n+2)/3}) \). As proved in this paper, all these functions have a similar differential spectrum and the question of equivalence of these functions can be raised. As the CCZ and the EA equivalence [9,8] preserve the Walsh spectrum, experiments have been done to compute the non-linearity and the Walsh spectrum of the functions \( G_t \) in \( \mathbb{F}_{2^n} \) for \( n < 18 \). While
for small \( n \) some functions have the same Walsh spectrum, this property is not true anymore for larger \( n \). Among the different properties observed when computing the non-linearity, we notice that two symmetric functions \( G_t \) and \( G_s \) with related differential spectrum do not necessary have the same non-linearity. These experimental results show that in general the functions are not affine equivalent.

Notice that the function \( G_s \) with \( s = \frac{n+3}{2} \) when \( n \equiv \pm 1 \) mod 6 has large algebraic degree and its inverse too. As it is well known that if the degree of the function is too small or too large the cipher can be sensitive to algebraic attacks, this function can be a relatively good candidate for the S-box of a block cipher.

Simulation over monomials \( G_t \) for \( (n \leq 31) \) shows that almost all functions \( G_t \) locally differentially 6-uniform are of a form in Table 1. From the simulation of [5], we have argument to conjecture that for \( n > 16 \), all power functions differentially 6-uniform are of the form \( x \mapsto x^{2^t} \). If both of these conjectures are true, the classification of differentially 6-uniform power functions is almost complete.

### Table 1: Differential Uniformity and Algebraic degree of the function \( G_t(x) = x^{2^t} \), their symmetric \( G_s \) and their inverses \( G_t^{-1} \) and \( G_s^{-1} \). We have \( \Delta = \max_{b \neq 0,1} \delta(b) \) and \( 1 \leq k, j \leq 2 \).

<table>
<thead>
<tr>
<th>( t,\ \deg(G_t) )</th>
<th>( s,\ \deg(G_s) )</th>
<th>( \deg(G_t^{-1}) )</th>
<th>( \deg(G_s^{-1}) )</th>
<th>( \Delta )</th>
<th>( \delta(G_t) )</th>
<th>( \delta(G_s) )</th>
<th>Spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 ( \frac{n+1}{2} )</td>
<td>( n-1 )</td>
<td>( \frac{n+1}{2} )</td>
<td>( n-1 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>(2,4)</td>
</tr>
<tr>
<td>( \frac{n}{2} + 1 )</td>
<td>( \frac{n+1}{2} )</td>
<td>*</td>
<td>( \frac{n+2}{2} )</td>
<td>2</td>
<td>( 2^{n/2-2} )</td>
<td>( 2^{n/2} )</td>
<td>[5]</td>
</tr>
<tr>
<td>3 ( \frac{n-2}{3} )</td>
<td>( \frac{n+1}{3} )</td>
<td>*</td>
<td>( \frac{n-1}{3} )</td>
<td>6</td>
<td>6</td>
<td>(6,8)</td>
<td>[5]</td>
</tr>
<tr>
<td>( \frac{kn+1}{3} )</td>
<td>( \frac{(3-k)n+2}{3} )</td>
<td>3</td>
<td>( \frac{(3-k)n+3}{3} )</td>
<td>( \frac{6}{3} )</td>
<td>6</td>
<td>( 2^{n/3-2} )</td>
<td>( 2^{n/3} )</td>
</tr>
</tbody>
</table>

### 6 Conclusion

Studying the properties of power functions is of great interest for the security of symmetric cryptographic primitives. As the differential spectrum of known families of APN and differentially 4-uniform power functions have already been studied, further investigations on power functions lead naturally to the study of the differentially 6-uniform ones. In [5], it was conjectured that a large number of differentially 6-uniform power functions are such that \( G_t(x) = x^{2^t} \) in \( \mathbb{F}_{2^n}[X] \). While the differential spectrum when \( t = 3 \) and \( t = n-2 \) was already presented in [5], in this paper we present the differential spectrum of the functions \( G_t \) for other values of \( t: t = \frac{n-1}{2}, \ t = \frac{n+3}{2}, \ t = \frac{kn+1}{3} \) and \( t = \frac{(3-k)n+2}{3} \) when \( t \) is an integer value and \( k = 1, 2 \). While these differential spectra are similar to the one of the function \( x^7 \), the algebraic degrees of some of these functions and of their inverses can provide better candidate for the S-boxes of a block cipher.
Acknowledgements

The authors wish to thank the anonymous reviewers for helpful comments and would like to thank Pascale Charpin for the advices provided when writing this article. The work of Léo Perrin is partly supported by an Erasmus Thesis scholarship.

References

Planar functions over fields of characteristic two
(Extended Abstract)

Kai-Uwe Schmidt · Yue Zhou

Received: 21 December 2012 / Accepted: 01 February 2013

Abstract Classical planar functions are functions from a finite field to itself and give rise to finite projective planes. They exist however only for fields of odd characteristic. We study their natural counterparts in characteristic two, which we also call planar functions. They again give rise to finite projective planes, as recently shown by the second author. We give a characterisation of planar functions in characteristic two in terms of codes over $\mathbb{Z}_4$. We then specialise to planar monomial functions $f(x) = cx^t$ and present constructions and partial results towards their classification. In particular, we show that $t = 1$ is the only odd exponent for which $f(x) = cx^t$ is planar (for some nonzero $c$) over infinitely many fields. The proof techniques involve methods from algebraic geometry.

Keywords planar functions · algebraic curves · finite fields · codes over $\mathbb{Z}_4$

1 Introduction

A function $f : \mathbb{F}_q \to \mathbb{F}_q$ is planar if

$$x \mapsto f(x + \epsilon) - f(x)$$

is a permutation of $\mathbb{F}_q$ for each $\epsilon \in \mathbb{F}_q^*$. Planar functions have been introduced by Dembowski and Ostrom [3] to construct finite projective planes and arise in

K.-U. Schmidt was partially supported by German Research Foundation.

Kai-Uwe Schmidt
Faculty of Mathematics, Otto-von-Guericke University
Universitätsplatz 2, 39106 Magdeburg, Germany
E-mail: kaiuwe.schmidt@ovgu.de

Yue Zhou
Faculty of Mathematics, Otto-von-Guericke University
Universitätsplatz 2, 39106 Magdeburg, Germany
E-mail: yue.zhou@st.ovgu.de
many other contexts. For example, Ganley and Spence [7] showed that planar functions give rise to certain relative difference sets, Nyberg and Knudsen [19], among others, studied planar functions (under the synonym perfect nonlinear functions) for applications in cryptography, and Carlet, Ding, and Yuan [2], among others, used planar functions to construct error-correcting codes.

Planar functions cannot exist in characteristic two since, if $q$ is even and $x$ is a solution to $f(x + \epsilon) - f(x) = a$ for $a \in \mathbb{F}_q$, then so is $x + \epsilon$. This is the motivation to define a function $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$ to be almost perfect nonlinear if (1) is a 2-to-1 map. Such functions have also been studied extensively for applications in cryptography and coding theory (see Carlet, Charpin, and Zinoviev [1], for example). However, there is no apparent link between almost perfect nonlinear functions and finite projective planes.

Recently, the second author proposed [22] a concept to overcome the problem that there is no planar function in characteristic two. The definition of a planar function has to be modified as follows.

**Definition 1** A function $f : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is planar if

$$x \mapsto f(x + \epsilon) + f(x) + \epsilon x$$

is a permutation of $\mathbb{F}_{2^n}$ for each $\epsilon \in \mathbb{F}_{2^n}^*$. Such functions share many of the properties of planar functions in odd characteristic. In Section 2 we provide further background on planar functions in characteristic two and discuss connections to finite geometries and coding theory. In Section 3 we specialise to monomial mappings, where we provide constructions of planar monomial mappings and present partial results towards their classification.

Full proofs and further details are given in a preliminary full version of this paper [21].

### 2 Background and Motivation

#### 2.1 Relative difference sets and finite geometries

Let $G$ be a finite group and let $N$ be a subgroup of $G$. A subset $D$ of $G$ is a relative difference set with parameters $(|G|/|N|, |N|, |D|, \lambda)$ and forbidden subgroup $N$ if the list of nonzero differences of $D$ comprises every element in $G \setminus N$ exactly $\lambda$ times. We are interested in relative difference sets $D$ with parameters $(q, q, q, 1)$ and a normal forbidden subgroup, in which case a classical result due to Ganley and Spence [7, Theorem 3.1] shows that $D$ can be uniquely extended to a finite projective plane.

It is known [6], [15] that, for even $q$, a relative difference set with parameters $(q, q, q, 1)$ in an abelian group necessarily satisfies $q = 2^n$ for some integer $n$ and is a subset of $\mathbb{Z}_4^n$ (where $\mathbb{Z}_4 = \mathbb{Z}/4\mathbb{Z}$) and the forbidden subgroup is $2\mathbb{Z}_4^n$. This fact was the motivation for the second author to study [22] such relative
difference sets, which then led to the notation of planar functions over fields of characteristic two.

We shall follow an approach that is slightly different from that in [22] and identify $\mathbb{Z}_4^n$ with the additive group of the Galois ring $R_n$ of characteristic 4 and cardinality $4^n$. We recall some basic facts about such Galois rings (see [18] or [9], for example). The unit group $R_n \setminus 2R_n$ of $R_n$ contains a cyclic subgroup $\Gamma(R_n)^*$ of size $2^n - 1$ and $\Gamma(R_n) = \Gamma(R_n)^* \cup \{0\}$ is called the Teichmuller set in $R_n$. We define addition on $\Gamma(R_n)$ by

$$x \oplus y = x + y + 2\sqrt{xy}$$

(3)

(where + is addition in $R_n$). Then $(\Gamma(R_n), \oplus, \cdot)$ is a finite field with $2^n$ elements [18, Statement 2]. Every $y \in R_n$ can be written uniquely in the form $y = a + 2b$ for $a, b \in \Gamma(R_n)$.

It is now an easy exercise to show that a relative difference set in $R_n$ with parameters $(2^n, 2^n, 2^n, 1)$ can always be written as

$$D = \{x + 2\sqrt{f(x)} : x \in \Gamma(R_n)\},$$

(4)

where $f$ is some function from $\Gamma(R_n)$ to itself. The following result characterises the functions $f$ for which (4) is a relative difference set.

**Theorem 2** The set $D$, given in (4), is a relative difference set with parameters $(2^n, 2^n, 2^n, 1)$ and forbidden subgroup $2R_n$ if and only if $f$ is planar.

Theorem 2 is essentially equivalent to [22, Theorem 2.1], which avoids using Galois rings at the cost of a more delicate proof.

2.2 Coding theory

We assume that the reader is familiar with the basic terminology of coding theory, in particular of the theory of codes over $\mathbb{Z}_4$. Otherwise, we advise to consult the seminal paper [9].

Let $f$ be a function from $F_2^n$ to itself satisfying $f(0) = 0$ and let $\alpha$ be a generator of $F_2^n$. It is well known (see [1, Theorem 5], for example) that for $n \geq 4$ the code over $F_2$ having parity check matrix

$$\begin{bmatrix}
1 & \alpha & \alpha^2 & \cdots & \alpha^{2^n-2} \\
1 & f(\alpha) & f(\alpha^2) & \cdots & f(\alpha^{2^n-2})
\end{bmatrix}$$

(5)

has minimum (Hamming) distance 3, 4, or 5, where the value 5 occurs if and only if $f$ is almost perfect nonlinear. We shall provide a similar characterisation for planar functions in characteristic two.

Let $f$ be a function from $\Gamma(R_n)$ to itself and let $\beta$ be a generator of $\Gamma(R_n)^*$. Consider the code $C_f$ over $\mathbb{Z}_4$ having parity check matrix

$$\begin{bmatrix}
1 & 1 & \alpha & \beta & \beta^2 & \cdots & \beta^{2^n-2} & 1 \\
2\sqrt{f(0)} & 1 & 2\sqrt{f(1)} & \beta & 2\sqrt{f(\beta)} & \beta^2 & \cdots & 2\sqrt{f(\beta^{2^n-2})} & 1
\end{bmatrix}.$$
This code and its dual are free $\mathbb{Z}_4$-modules of rank $4^{2^n-n-1}$ and $4^{n+1}$, respectively.

We remind the reader that the Lee weights of $0, 1, 2, 3 \in \mathbb{Z}_4$ are $0, 1, 2, 1$, respectively, and the Lee weight $wt_L(c)$ of $c \in (\mathbb{Z}_4)^N$ is the sum of the Lee weights of its components. This weight function defines a metric in $(\mathbb{Z}_4)^N$, called the Lee distance.

Write $\mathcal{C}$ for the code $\mathcal{C}_f$ when $f$ is identically zero (in which case $f$ is planar). The dual code $\mathcal{C}^\perp$ is the $\mathbb{Z}_4$-Kerdock code described in [9]. Let

$$\phi : (\mathbb{Z}_4)^N \rightarrow (\mathbb{F}_2)^{2N}$$

be the Gray map, which defines an isometry between $(\mathbb{Z}_4)^N$, equipped with the Lee distance, and $(\mathbb{F}_2)^{2N}$, equipped with the Hamming distance. Then, for $n \geq 3$ odd, $\phi(\mathcal{C}^\perp)$ is the classical Kerdock code and $\phi(\mathcal{C})$ has the same parameters as the Preparata code (see [9] for details on these codes).

The Lee weight distribution of $\mathcal{C}^\perp$ has been determined in [9]. The following result gives a characterisation of planar functions.

**Theorem 3** The code $(\mathcal{C}_f)^\perp$ has the same Lee weight distribution as $\mathcal{C}^\perp$ if and only if $f$ is planar.

For odd $n$, we have the following alternative characterisations of planar functions.

**Theorem 4** For odd $n \geq 3$, the code $\mathcal{C}_f$ has minimum Lee distance 4 or 6, where the value 6 occurs if and only if $f$ is planar.

A consequence of Theorem 4 is the following.

**Corollary 5** For odd $n \geq 3$, the code $\phi(\mathcal{C}_f)$ punctured in one (arbitrary) coordinate has minimum distance 3, 4, or 5, where the value 5 occurs if and only if $f$ is planar.

Let $\mathcal{D}_f$ be the code over $\mathbb{F}_2$ with parity check matrix (5). If $f$ is almost perfect nonlinear and $n \geq 4$, then $\mathcal{D}_f$ has parameters $(2^n - 1, 2^{2n-2n-1}, 5)$. In contrast, by Corollary 5, if $f$ is planar and $n \geq 3$ is odd, then $\phi(\mathcal{C}_f)$ punctured in one coordinate has parameters $(2^n - 1, 2^{2n-2n}, 5)$, and so contains twice as many codewords as $\mathcal{D}_f$. If $f$ is planar, then $\phi(\mathcal{C}_f)$ punctured in one coordinate meets a version of the Johnson bound, and so is nearly perfect [8].

3 Planar monomial functions

Every function from $\mathbb{F}_{2^n}$ to itself can be uniquely written as a polynomial function of degree at most $2^n - 2$. We consider the simplest nontrivial polynomial functions, namely monomial functions $x \mapsto cx^t$ for some $c \in \mathbb{F}_{2^n}^*$ and some integer $t$. Such functions are often preferred in applications. We are interested in those exponents $t$ that give rise planar functions.

**Definition 6** An integer $t$ satisfying $1 \leq t \leq 2^n - 2$ is a planar exponent of $\mathbb{F}_{2^n}$ if the function $x \mapsto cx^t$ is planar on $\mathbb{F}_{2^n}$ for some $c \in \mathbb{F}_{2^n}^*$. 
Trivially, $2^k$ is a planar exponent of all fields $\mathbb{F}_{2^n}$ satisfying $n > k$. A nontrivial example is given in the following theorem, in which $\text{Tr}_m(x) = x + x^2 + \cdots + x^{2^m-1}$ denotes the absolute trace function on $\mathbb{F}_{2^m}$.

**Theorem 7** Let $c \in \mathbb{F}_{4^k}^*$ be such that $\text{Tr}_k(c) = 0$. Then the function

$$x \mapsto cx^{2^k+1}$$

is planar on $\mathbb{F}_{4^k}$.

**Remark** After receiving a preprint of our work [21], Scherr and Zieve [20] characterised all $c \in \mathbb{F}_{4^k}$ such that $x \mapsto cx^{2^k+1}$ is planar on $\mathbb{F}_{4^k}$.

Theorem 7 shows that $2^k + 1$ is a planar exponent of $\mathbb{F}_{4^k}$. In a preprint of this paper [21], we conjectured that $4^k(4^k + 1)$ is a planar exponent of $\mathbb{F}_{64^k}$. This was subsequently proved by Scherr and Zieve [20], by providing the following more precise result.

**Theorem 8 (Scherr and Zieve [20])** Let $c \in \mathbb{F}_{64^k}^*$ be a $(4^k - 1)$-th power but not a $3(4^k - 1)$-th power. Then the function

$$x \mapsto cx^{4^k(4^k+1)}$$

is planar on $\mathbb{F}_{64^k}$.

We conjecture that the above examples of planar exponents form in fact the complete list of planar exponents.

**Conjecture 9** If $t$ is a planar exponent of $\mathbb{F}_{2^n}$, then $t$ is one of the values given in Table 1.

The following partial answer to Conjecture 9 is easy to prove.

**Proposition 10** Let $t$ be an integer satisfying $\gcd(t - 2, 2^n - 1) = 1$. If $t$ is a planar exponent of $\mathbb{F}_{2^n}$, then $t$ is a power of 2.

As in odd characteristic, the classification of planar monomials in characteristic two seems to be a challenging problem. This motivates us to study the relaxed problem of classifying those numbers that are planar exponents of $\mathbb{F}_{2^n}$ for infinitely many $n$. The only known such numbers are the powers of 2 and we have the following weaker form of Conjecture 9.

### Table 1 Conjectured complete list of planar exponents of $\mathbb{F}_{2^n}$

<table>
<thead>
<tr>
<th>Exponent $t$</th>
<th>Condition</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^k$</td>
<td>none trivial</td>
<td></td>
</tr>
<tr>
<td>$2^k + 1$</td>
<td>$n = 2k$</td>
<td>Theorem 7</td>
</tr>
<tr>
<td>$4^k(4^k + 1)$</td>
<td>$n = 6k$</td>
<td>Theorem 8</td>
</tr>
</tbody>
</table>
Conjecture 11 If $t$ is a planar exponent of $\mathbb{F}_{2^n}$ for infinitely many $n$, then $t$ is a power of 2.

Our main result is the following.

Theorem 12 If $t$ is an odd planar exponent of $\mathbb{F}_{2^n}$ for infinitely many $n$, then $t = 1$.

The problem of classifying the numbers that are planar exponents of $\mathbb{F}_{2^n}$ for infinitely many $n$ parallels the problem of classifying monomial functions $x \mapsto x^t$ on $\mathbb{F}_{2^n}$ that are almost perfect nonlinear for infinitely many $n$. To attack this problem, Janwa, McGuire, and Wilson [13] proposed to use ideas from algebraic geometry. These ideas were further developed by Jedlicka [14] and Hernando and McGuire [10], leading to a complete solution. The same approach has been used by Hernando and McGuire [11] to prove a conjecture on monomial hyperovals in projective planes and by Leducq [16] and Hernando, McGuire, and Monserrat [12] to give partial results towards a classification of monomial functions $x \mapsto x^t$ on $\mathbb{F}_p^n$ (with $p$ odd) that are planar for infinitely many $n$ (which was recently completed by Zieve [23] using different techniques). We use a similar approach to prove Theorem 12, though our proof requires several extra ideas. The approach is described in the remainder of this section.

Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be of the form $f(x) = cx^t$ for some $c \in \mathbb{F}_{2^n}^*$ and let $\epsilon \in \mathbb{F}_{2^n}^*$. Then the condition that (2) is a permutation is equivalent to the condition that the polynomial

$$c(U + \epsilon)^t + c(V + \epsilon)^t + cU^t + cV^t + \epsilon(U + V)$$

has no zeros $(u, v)$ over $\mathbb{F}_{2^n}$ satisfying $u \neq v$. Substituting $U = \epsilon X$ and $V = \epsilon Y$, we see that this condition is in turn equivalent to the condition that the polynomial

$$(X + 1)^t + (Y + 1)^t + X^t + Y^t + a(X + Y)$$

(6)

has no zeros $(u, v)$ over $\mathbb{F}_{2^n}$ satisfying $u \neq v$, where $a = \epsilon^{2^{-t}/c}$. The polynomial (6) is divisible by $X + Y$. We are therefore interested in the zeros of the polynomial

$$F_{t,a}(X, Y) = \frac{(X + 1)^t + (Y + 1)^t + X^t + Y^t + a(X + Y)}{X + Y}$$

(7)

(which however could still have zeros on the line $X + Y$). We consider the affine plane curve defined by $F_{t,a}$ (and follow the usual convention to denote the curve and a defining polynomial by the same symbol). Then, defining a subset of $\mathbb{F}_{2^n}$ by

$$A_n = \{ \epsilon^{2^{-t}/c} : \epsilon \in \mathbb{F}_{2^n}^* \},$$

(8)

the function $x \mapsto cx^t$ is planar on $\mathbb{F}_{2^n}$ if and only if the curve $F_{t,a}$ has rational points $(u, v)$ over $\mathbb{F}_{2^n}$ satisfying $u \neq v$ for some $a \in A_n$. 
The number of rational points on a curve can be estimated using Weil’s Theorem (see [4, Theorem 5.4.1], for example). Roughly speaking, Weil’s Theorem asserts that, if \( F \in \mathbb{F}_q[X,Y] \) has an absolutely irreducible factor and \( q \) is large compared to the degree of \( F \), then the number of rational points over \( \mathbb{F}_q \) on the affine plane curve defined by \( F \) is approximately \( q \). A simple consequence of Weil’s Theorem is the following.

**Proposition 13** If \( F_{t,a} \) has an absolutely irreducible factor over \( \mathbb{F}_{2^n} \) for some \( a \neq 1 \) in \( \mathcal{A}_n \) and \( n \) is sufficiently large, then \( t \) is not a planar exponent of \( \mathbb{F}_{2^n} \).

In view of Proposition 13, Conjecture 11 is proved by showing that, when \( t \) is not a power of 2, \( F_{t,a} \) has an absolutely irreducible factor over \( \mathbb{F}_{2^n} \) for some \( a \in \mathcal{A}_n \) satisfying \( a \neq 1 \) and all sufficiently large \( n \). We indeed prove this when \( t > 1 \) is odd, and so prove Theorem 12.

The case that \( t = 2^k + 1 \) is particularly easy to handle.

**Proposition 14** Let \( t \) be a number of the form \( 2^k + 1 \) for integral \( k \geq 1 \). Then \( F_{t,a} \) has an absolutely irreducible factor for some \( a \neq 1 \) in \( \mathcal{A}_n \) and for all sufficiently large \( n \).

**Proof** Notice that \( F_{t,a} \) simplifies to

\[
F_{t,a}(X,Y) = (X + Y)^{2^k-1} + a + 1.
\]

We claim that, for all sufficiently large \( n \), we can choose \( a \neq 1 \) in \( \mathcal{A}_n \) such that

\[
a + 1 = b^{2^k-1}.
\]

for some \( b \in \mathbb{F}_{2^n} \). This will prove the proposition since then \( X + Y + b \) divides \( F_{t,a} \). By the definition (8) of \( \mathcal{A}_n \), the claim is equivalent to the existence of \( \epsilon, b \in \mathbb{F}_{2^n}^* \) such that, for all \( c \in \mathbb{F}_{2^n}^* \),

\[
\epsilon^{1-2^k} / c + 1 = b^{2^k-1}, \tag{9}
\]

which in turn is equivalent to

\[
\epsilon^{2^k-1} + x^{2^k-1} = 1 / c, \tag{10}
\]

where \( x = cb \). It is well known [17, Example 6.38] that the number of solutions \( (\epsilon, x) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \) to the equation (10) is at least

\[
2^n - (2^k - 2)(2^k - 3)2^{n/2} - 2^k + 2.
\]

Since there are at most \( 2^k - 1 \) solutions of the form \((0, x)\) and at most \( 2^k - 1 \) solutions of the form \((\epsilon, 0)\), we find that, for all sufficiently large \( n \), there exist \( \epsilon, x \in \mathbb{F}_{2^n}^* \) satisfying (10). Hence, for all sufficiently large \( n \), there exist \( \epsilon, b \in \mathbb{F}_{2^n}^* \) satisfying (9), as required. \( \square \)
Now let $t = 2^k \ell + 1$ for integers $k \geq 1$ and odd $\ell \geq 3$. In this case, our proof of Theorem 12 is much more involved. Instead of looking at $F_t$ directly, we consider its homogenised version $H_t(X, Y, Z)$, namely

$$H_t(X, Y, Z) = \frac{(X + Z)^t + (Y + Z)^t + X^t + Y^t + a(X + Y)Z^{t-1}}{Z(X + Y)}. \quad (11)$$

Of course, $F_t$ has an absolutely irreducible factor if and only if $H_t$ has an absolutely irreducible factor. Our strategy is to consider the projective plane curve defined by $H_t$ over the algebraic closure $\mathbb{F}$ of $\mathbb{F}_2$ and derive a contradiction to Bezout’s Theorem (see [5, § 5.3], for example) under the assumption that $H_t$ has no absolutely irreducible factor over $\mathbb{F}_2$.

**Bezout’s Theorem.** Let $A$ and $B$ be two projective plane curves over an algebraically closed field $K$, having no component in common. Then

$$\sum_P I_P(A, B) = (\deg A)(\deg B), \quad (12)$$

where the sum runs over all points in the projective plane $\mathbb{P}^2(K)$.

Notice that $I_P(A, B)$ is the intersection number of $A$ and $B$ at $P$, whose precise definition is neither recalled nor required here. We derive the desired contradiction to Bezout’s Theorem by factoring $H_t$ into putative factors $A$ and $B$ over some extension of $\mathbb{F}_2$ and computing the intersection numbers $I_P(A, B)$ at the singular points of $H_t$. After obtaining upper bounds for the number of singular points of $H_t$ of certain types, we conclude that the left hand side of (12) can never be as big as the right hand side of (12).

**References**

An upper bound on the size of Kakeya sets in finite vector spaces

Gohar Kyureghyan · Qi Wang

Received: date / Accepted: date

Abstract For a finite field $\mathbb{F}_q$, a Kakeya set $K$ is a subset of $\mathbb{F}_q^n$ that contains a line in every direction. This paper derives an upper bound on the minimum size of Kakeya sets for even $q$.

Keywords Kakeya set · finite vector space · Gold power function

Mathematics Subject Classification (2010) 11T30 · 11T06

1 Introduction

Let $\mathbb{F}_q$ be a finite field with $q$ elements. A Kakeya set $K \subset \mathbb{F}_q^n$ is a set containing a line in every direction. More formally, $K \subset \mathbb{F}_q^n$ is a Kakeya set if and only if for every $x \in \mathbb{F}_q^n$, there exists $y \in \mathbb{F}_q^n$ such that \( \{y + tx : t \in \mathbb{F}_q\} \subset K \).

Wolff in [11] asked whether a lower bound of the form $|K| \geq C_n \cdot q^n$ holds for all Kakeya sets $K$, where $C_n$ is a constant depending only on $n$. Dvir [3] first gave such a lower bound with $|K| \geq (1/n!)q^n$. Later Dvir, Kopparty, Saraf and Sudan improved the lower bound to $|K| \geq (1/2^n)q^n$ in [5] (see also [10]). It was shown in [5] that for any $n \geq 1$ there exists a Kakeya set $K \subset \mathbb{F}_q^n$ with

$$|K| \leq 2^{-(n-1)}q^n + O(q^{n-1}).$$

(1)

For more information on Kakeya sets, we refer to a recent survey [4].

When $q$ is bounded and $n$ grows, the bound above is weak, and some recent papers improved the $O$-term in (1) to have a better upper bound for this case.
The best currently known such bound was obtained by Kopparty, Lev, Saraf and Sudan in [7], following the ideas from [10, 5] (see also [9]):

**Theorem 1** [7, Theorem 6] Let \( n \geq 1 \) be an integer and \( q \) a prime power. There exists a Kakeya set \( K \subset \mathbb{F}_q^n \) with

\[
|K| < \begin{cases} 
2 \left( 1 + \frac{1}{q-1} \right) \left( \frac{2q+1}{2} \right)^n & \text{if } q \text{ is odd}, \\
\frac{3}{2} \left( 1 + \frac{1}{q^{1/2}-1} \right) \left( \frac{2q+1}{4} \right)^n & \text{if } q \text{ is an even power of 2}, \\
\frac{3}{2} \left( \frac{2(q+\sqrt{q}+1)}{3} \right)^n & \text{if } q \text{ is an odd power of 2}.
\end{cases}
\]

Theorem 1 was proved by constructing a Kakeya set \( K \subset \mathbb{F}_q^n \) from a suitable function \( f : \mathbb{F}_q \to \mathbb{F}_q \). Let \( t \in \mathbb{F}_q \) and set \( I_f(t) := \{ f(x) + tx : x \in \mathbb{F}_q \} \). Further, define

\[
K := \{ (x_1, \ldots, x_j, t, 0, \ldots, 0) : 0 \leq j \leq n-1, t \in \mathbb{F}_q, x_1, \ldots, x_j \in I_f(t) \}.
\]

If \( f \) is a non-linear function, then \( K \) is indeed a Kakeya set [7] of size

\[
|K| = \sum_{j=0}^{n-1} \sum_{t \in \mathbb{F}_q} |I_f(t)|^j = \sum_{t \in \mathbb{F}_q} \frac{|I_f(t)|^n - 1}{|I_f(t)| - 1}.
\]  

(2)

Clearly, to construct a small Kakeya set, we need to find a function \( f : \mathbb{F}_q \to \mathbb{F}_q \) for which the sets \( I_f(t) \) are small. Theorem 1 was obtained by taking

- \( f(x) = x^2 \) for \( q \) odd, since then \( |I_f(t)| \leq (q+1)/2 \) holds for all \( t \in \mathbb{F}_q \);
- \( f(x) = x^3 \) for \( q \) an even power of 2, since then \( |I_f(t)| \leq (2q+1)/3 \) holds for all \( t \in \mathbb{F}_q \);
- \( f(x) = x^{q-2} + x^2 \) for \( q \) an odd power of 2, since then \( |I_f(t)| \leq 2(q+\sqrt{q}+1)/3 \) holds for all \( t \in \mathbb{F}_q \).

In [7], it was also mentioned that it might be possible to choose better non-linear functions \( f : \mathbb{F}_q \to \mathbb{F}_q \) to improve the bounds in Theorem 1. In this paper, we investigate this idea further and derive indeed a better upper bound on the minimum size of Kakeya sets \( K \subset \mathbb{F}_q^n \). More precisely, we show that the Gold power functions allow to improve the upper bounds in Theorem 1 for \( q \) even. Gold power functions are intensively studied because of their various applications in coding theory and cryptography [6, 1, 2]. Our main result is

\[
|K| < \frac{2q}{q + \sqrt{q} - 2} \left( \frac{q + \sqrt{q}}{2} \right)^n,
\]  

(3)

where \( q \) is an even power of 2.
2 New upper bounds on the size of Kakeya sets

In this section, we use the Gold power functions to derive the upper bounds on the size of Kakeya sets \( K \subset \mathbb{F}_q^n \) with \( q \) even. The key tool for obtaining better bounds is the following result by Bluher [1]:

**Theorem 2** [1, Theorem 5.6] Let \( q = 2^m, f(x) = x^{2^i+1} \in \mathbb{F}_q[x] \) with \( 0 \leq i < m \), and \( d = \gcd(i, m) \). Let \( N_0 \) denote the number of \( b \in \mathbb{F}_q^* \) such that \( f(x) + bx + b \) has no root in \( \mathbb{F}_q \). On \( N_0 \) we have:

(i) if \( m/d \) is even, then \( N_0 = \frac{2^d(q-1)}{2(2^d+1)} \);

(ii) if \( m/d \) is odd, then \( N_0 = \frac{2^d(q+1)}{2(2^d+1)} \).

With the result above, we can determine the size of the image set \( I_f(t) := \{ f(x) + tx : x \in \mathbb{F}_q \} \) with \( f(x) = x^{2^i+1} \) and \( t \in \mathbb{F}_q^* \).

**Proposition 1** Let \( q = 2^m, f(x) = x^{2^i+1} \in \mathbb{F}_q[x] \) with \( 0 \leq i < m \), and \( d = \gcd(i, m) \). Set \( I_f(t) := \{ f(x) + tx : x \in \mathbb{F}_q \} \) for \( t \in \mathbb{F}_q^* \). We have:

(i) if \( m/d \) is even, then \( |I_f(0)| = 1 + \frac{q-1}{2^d+1} \), and \( |I_f(t)| = q + \frac{q-1}{2(2^d+1)} \) for any \( t \in \mathbb{F}_q^* \);

(ii) if \( m/d \) is odd, then \( |I_f(0)| = q \), and \( |I_f(t)| = q + \frac{q+1}{2(2^d+1)} \) for any \( t \in \mathbb{F}_q^* \).

**Proof** For \( t = 0 \), we have

\[
|I_f(0)| = 1 + \frac{2^m-1}{\gcd(2^m-1, 2^i+1)}.
\]

From the well-known fact (e.g. [8, Lemma 11.1]) that

\[
\gcd(2^m-1, 2^i+1) = \begin{cases} 
1 & \text{if } m/d \text{ is odd,} \\
2^d+1 & \text{if } m/d \text{ is even,}
\end{cases}
\]

the assertion on \( |I_f(0)| \) follows.

For \( t \in \mathbb{F}_q^* \), by definition, we have

\[
|I_f(t)| = |\{ f(x) + tx : x \in \mathbb{F}_q \}| = |\mathbb{F}_q| - |\{ c \in \mathbb{F}_q^*: f(x) + tx + c \text{ has no root in } \mathbb{F}_q \}| = q - N'_0.
\]

To make use of Theorem 2, we transform \( f(x) + tx + c \) as follows. Since \( t \neq 0 \) and \( c \neq 0 \), let \( x = \frac{c}{t} z \), then

\[
f(x) + tx + c = x^{2^i+1} + tx + c = \frac{c^{2^i+1}}{t^{2^i+1}} \left( z^{2^i+1} + \frac{(2^i+1)}{c^{2^i}} z + \frac{t^{2^i+1}}{c^{2^i}} \right).
\]
Since
\[
\left\{ \frac{t^{2i+1}}{c^{2i}} : c \in \mathbb{F}_q^* \right\} = \mathbb{F}_q^*,
\]
we have \( N'_0 = N_0 \), where \( N_0 \) denotes the number of \( b \in \mathbb{F}_q^* \) such that \( x^{2i+1} + bx + b \) has no root in \( \mathbb{F}_q \). The conclusion then follows from Theorem 2. \( \square \)

Proposition 1 shows that the smallest Kakeya sets constructed using Gold power functions can be achieved if we take \( i = m/2 \) for an even \( m \), and \( i = 0 \) for an odd \( m \). The below discussion shows that the choice \( i = m/2 \) implies a better upper bound on Kakeya sets compared with the one given in Theorem 1. Observe that \( f(x) = x^3 \) chosen in [7] to prove the bound for \( m \) even is the Gold power function with \( i = 1 \) and \( d = 1 \).

When \( m/d \) is odd, \( |I_{f(0)}| = q \), and therefore the bound obtained by the Gold power functions cannot be better for large \( n \). However, for small values of \( n \), it is better than the one of Theorem 1.

**Theorem 3** Let \( q = 2^m \), \( f(x) = x^{2i+1} \in \mathbb{F}_q[x] \) with \( 0 \leq i < m \) and \( d = \gcd(i, m) \). If \( m \) is even and \( m/d \) is even, then \( f(x) \) yields a Kakeya set \( K \subset \mathbb{F}_q^n \) with
\[
|K| = \frac{2d+1}{q-1} \left( \left( \frac{q+2d}{2d+1} \right)^n - 1 \right) + \frac{2d+1}{2d+1} \left( \left( \frac{2dq + 2q + 2d}{2(2d+1)} \right)^n - 1 \right).
\]
In particular, by choosing \( i = m/2 \), there exists a Kakeya set \( K \subset \mathbb{F}_q^n \) with
\[
|K| = \frac{(\sqrt{q})^n - 1}{\sqrt{q} - 1} + \frac{2\sqrt{q} + 2}{\sqrt{q} + 2} \left( \left( \frac{q + \sqrt{q}}{2} \right)^n - 1 \right).
\]
If \( m \) is odd, then \( f(x) \) yields a Kakeya set \( K \subset \mathbb{F}_q^n \) with
\[
|K| = \frac{q^n - 1}{q - 1} + \frac{(2d+1)(q-1)}{2^{d-1}(q-3) + (q-1)} \left( \left( \frac{2dq + 2q - 2d}{2(2d+1)} \right)^n - 1 \right).
\]
In particular, by choosing \( i = 0 \), there exists a Kakeya set \( K \subset \mathbb{F}_q^n \) with
\[
|K| = \frac{q^n - 1}{q - 1} + \frac{2(q-1)}{q - 2} \left( \left( \frac{q}{2} \right)^n - 1 \right).
\]
**Proof** The statement follows from relation (2) and Proposition 1. \( \square \)

The Gold power functions \( f(x) = x^{2i+1} \) with \( 0 \leq i < m \) imply the following upper bounds on Kakeya sets \( K \subset \mathbb{F}_q^n \).

**Corollary 1** If \( m \) is even, then there exists a Kakeya set \( K \subset \mathbb{F}_q^n \) with
\[
|K| < \frac{(2d+1)q}{(2d+1)(q-1)} \left( \left( \frac{2dq + 2q + 2d}{2(2d+1)} \right)^n \right).
\]
where \( d = \gcd(i, m) \) and \( m/d \) is even. In particular, by choosing \( i = m/2 \), there exists a Kakeya set \( K \subset F_q^n \) with

\[
|K| < \frac{2q}{q + \sqrt{q} - 2} \left( \frac{q + \sqrt{q}}{2} \right)^n.
\]

(4)

If \( m \) is odd, then there exists a Kakeya set \( K \subset F_q^n \) with

\[
|K| < q^n - 1 \left( \frac{q - 1}{q - 2} \right) \left( \frac{q}{2} \right)^n.
\]

(5)

**Remark 1** Let \( q = 2^m \) with \( m \geq 2 \) even. Then the function chosen in Theorem 1 is the Gold power function with \( i = 1 \) and \( d = 1 \). Hence Corollary 1 always gives a better estimate on the size of Kakeya sets if \( d > 1 \) and \( m/d \) even. In particular, if we choose \( i = m/2 \), then we get the best bound (4) led by the Gold power functions. More precisely, in comparison, when \( q \) is fixed and \( n \) grows, the new upper bound is

\[
|K| < c_1 \left( \frac{q + \sqrt{q}}{2} \right)^n,
\]

where \( c_1 \) is a constant depending only on \( q \), which supersedes the bound in Theorem 1

\[
|K| < c_0 \left( \frac{2q + 1}{3} \right)^n,
\]

where \( c_0 \) is also a constant depending only on \( q \).

**Remark 2** For \( q = 2^m \) with \( m \geq 1 \) odd, the Gold power function \( f(x) = x^{2^i+1} \) is a permutation monomial in \( F_q \), and the size of the image set \( I_f(t) \) for \( t = 0 \) is \( |I_f(0)| = q \). We now compare the bound of (5) given by Corollary 1 with that in Theorem 1. For simplicity, let \( s_0 = 2(q + \sqrt{q} + 1)/3 \) and \( s_1 = \frac{q}{2} \). It is easily seen that \( s_1 < s_0 < q \) for any \( m \geq 1 \). Then by (2), the two bounds in Theorem 1 and Corollary 1 are

\[
|K| = \frac{s_0^n - 1}{s_0 - 1} \cdot q,
\]

\[
|K| = \frac{s_1^n - 1}{s_1 - 1} \cdot (q - 1) + \frac{q^n - 1}{q - 1} \cdot q - 1,
\]

respectively. Assume that the bound in (5) is better than the bound in Theorem 1, i.e.,

\[
\frac{s_1^n - 1}{s_1 - 1} \cdot (q - 1) + \frac{q^n - 1}{q - 1} \leq \frac{s_0^n - 1}{s_0 - 1} \cdot q.
\]

We then get

\[
\frac{q^n - 1}{q - 1} \leq \frac{s_1^n - 1}{s_1 - 1} \leq \left( \frac{s_0^n - 1}{s_0 - 1} \cdot \frac{s_1^n - 1}{s_1 - 1} \right) \cdot q.
\]

(6)

If \( q \) is bounded, in (6) the left-hand side increases in \( n \) faster than the right-hand side. Thus, there must exist some integer \( n_0 \), such that only for \( n \leq n_0 \),
the bound of (5) is better than that in Theorem 1. On the other hand, the inequality (6) holds if
\[
\frac{q^{n-1} - s_1^{n-1}}{s_0^{n-1} - s_1^{n-1}} \leq q.
\]
Note that
\[
\left( \frac{q}{s_0} \right)^{n-1} \left( \frac{\ln q}{\ln s_0} \right) \leq q.
\]
Therefore, if \( n < \left\lfloor \frac{\ln q}{\ln s_0} \right\rfloor + 1 \), the new bound in Corollary 1 is superior to that in Theorem 1. Furthermore, \( \left\lfloor \frac{\ln q}{\ln s_0} \right\rfloor \to \infty \) when \( q \to \infty \), which means that our bound in Corollary 1 for \( m \) odd is better for large \( n \) if \( q \) is large enough.

3 Conclusion

Using the Gold power functions \( f(x) = x^{2^i+1} \), we presented an upper bound on the minimum size of Kakeya sets \( K \subset \mathbb{F}_q \) with \( q = 2^m \). For \( m \) is even, by choosing \( i = m/2 \), we obtained a better bound than all previously known bounds. For odd \( m \), we argued that when \( q \) is bounded, the upper bound is still superior for \( n \leq n_0 \), where \( n_0 \) is an integer depending only on \( q \).

When \( m \) is odd, we remark that it is possible to choose other types of non-linear functions to derive better upper bound. For example, our numerical results for \( m \leq 11 \) show that the binomial \( f(x) = x^{13} + x^2 \) implies a better bound than that in Theorem 1.

Acknowledgment

Qi Wang’s research is supported by the Alexander von Humboldt (AvH) Stiftung/Foundation. Vsevolod F. Lev informed us that the upper bound (3) was proved also by Petter Müller (see http://mathoverflow.net/questions/102751/a-mixing-property-for-finite-fields-of-characteristic-2).

References

$q$-ary lattices in the $l_p$ norm and a generalization of the Lee metric

Grasiele C. Jorge · Antonio Campello · Sueli I. R. Costa

Received: date / Accepted: date

Abstract $q$-ary lattices are obtained from linear $q$-ary codes via the so-called Construction A. We study these lattices in the $l_p$ norm in $\mathbb{R}^n$ and the associated $q$-ary codes in the induced $p$-Lee metric in $\mathbb{Z}_q^n$. This induced metric extends to $1 \leq p < \infty$ the well-known Lee metric when $p = 1$. A previous result on lattice decoding in the Lee metric is generalized for $p > 1$ and the existence of perfect codes in such metrics is briefly discussed. In the case $p = \infty$ a complete characterization of perfect codes is given.

Keywords Lattices · $q$-ary codes · Lee metric

1 Introduction

One of the most used constructions of lattices from error-correcting codes is the so-called Construction A. It associates a lattice in $\mathbb{Z}^n$ to a linear code in $\mathbb{Z}_q^n$. Such lattices are also called $q$-ary lattices and have several applications in coding theory and cryptography. Lattice-based cryptographic schemes are usually built upon $q$-ary lattices and are linked to the computational difficulty of the shortest and closest vector problems (SVP and CVP) [8]. Construction A is also employed to build good codes for the Gaussian channel, as well as for some channels with side information [13].

There are not many references in the literature of lattices considered in the $l_p$ norm in $\mathbb{R}^n$ for $p \neq 2$. Peikert [9] studies the complexity of some important computational lattice problems such as the closest vector problem and the shortest vector problem in the $l_p$ norm for $2 \leq p \leq \infty$. In [6], Grell et al.

This work was partially supported by FAPESP 2011/22044-4, CNPq 150802/2012-9, 309561/2009-4

Grasiele C. Jorge, Antonio Campello and Sueli I. R. Costa
Institute of Mathematics, Statistics and Computer Science
University of Campinas, São Paulo, Brazil, 13083-859
E-mail: [grajorge, campello, sueli]@ime.unicamp.br
show optimal algorithms in these norms for the closest point search problem for some well-known families of lattices, such as $\mathbb{Z}_n$, $D_n$, $A_n$, $E_6$, $E_7$ and $E_8$.

In this paper, we consider integer lattices in the $l_p$ norm in $\mathbb{Z}_n$ and $q$-ary codes in the induced metric in $\mathbb{Z}_n^q$. We call this induced metric the $p$-Lee metric. It is worth noting that the $p$-Lee metric appeared in the literature as “generalized weights” of a code (cf. [10]). Also the “Euclidean weight” of a code, defined in [4] in the context of physical layer network coding, corresponds exactly to the 2-Lee metric in the sense of this paper.

In Section 4 it is shown the equivalence between the closest point search in a $q$-ary lattice in the $l_p$ norm and the closest point search in the underlying $q$-ary code in the induced $p$-Lee metric. In Section 5 is briefly discussed the existence of perfect codes in the $p$-Lee metric.

2 Codes and lattices

We consider here a $q$-ary linear code $C \subseteq \mathbb{Z}_q^n$ as a $\mathbb{Z}_q$-submodule of $\mathbb{Z}_q^n$, $q \in \mathbb{N}$. If $q$ is a prime number, then $C$ is a vector subspace of $\mathbb{Z}_q^n$ and therefore has a basis with $k \leq n$ vectors. Otherwise, we can only assure the existence of a minimal set of generators which are not necessarily linearly independent.

A lattice $\Lambda$ is a discrete additive subgroup of $\mathbb{R}^n$. Equivalently, $\Lambda \subseteq \mathbb{R}^n$ is a lattice iff there are linearly independent vectors $v_1, \ldots, v_m \in \mathbb{R}^n$ such that $\Lambda$ is the set of all integer linear combinations $\sum_{i=1}^{m} \alpha_i v_i$ with $\alpha_i \in \mathbb{Z}$. The set $\{v_1, \ldots, v_m\}$ is called a basis for $\Lambda$. A matrix $M$ whose columns are these vectors is said to be a generator matrix for $\Lambda$. The determinant of a full rank lattice $\Lambda$ ($m = n$), can be defined as $\det \Lambda = |\det M|$ and it is an invariant under basis change. A lattice $\Lambda$ is called integer if $\Lambda \subseteq \mathbb{Z}^n$.

Let $r \in \mathbb{R}^n$ be a received vector. An important problem regarding lattices, which has several applications in coding and cryptography, is finding a closest lattice vector to $r$ (lattice decoding). Considering the Euclidean distance this problem has been widely studied and has applications in information transmission over a channel with additive white Gaussian noise. This problem is computationally difficult for general integer lattices, proved to be NP-Hard [8].

The so-called Construction A associates to a $q$-ary code $C \subseteq \mathbb{Z}_q^n$ an integer lattice $\Lambda_A(C)$ via the surjective map $\phi$ [2]:

$$\phi : \mathbb{Z}_q^n \longrightarrow \mathbb{Z}_n^n$$

$$(x_1, \ldots, x_n)^t \longmapsto (\bar{x}_1, \ldots, \bar{x}_n)^t,$$

where $\bar{x}_i = x_i \mod q$ for $i = 1, \ldots, n$, $\Lambda_A(C) = \phi^{-1}(C)$. It is straightforward to see that $\Lambda_A(C)$ is a lattice if, and only if, $C$ is a linear code. In this case, $\Lambda_A(C)$ is called the $q$-ary lattice associated to $C$. Any $q$-ary lattice has $q\mathbb{Z}_n$ as a sublattice and this property can be used as an alternative definition to $q$-ary lattices, as is done in [8].

Some properties of Construction A are described in the following proposition.
Proposition 1 If $\Lambda_A(C)$ is the $q$-ary lattice associated to the code $C \subseteq \mathbb{Z}_q^n$, then:

1. $\left| \frac{A_A(C)}{q\mathbb{Z}^n} \right| = \frac{q^n}{\det A_A(C)} = |C|$, where $|C|$ is the number of codewords of $C$.

2. Any full rank integer lattice $\Lambda \subseteq \mathbb{Z}^n$ is $q$-ary for $q = \det(\Lambda)$.

Proof The first property is direct, due to the isomorphism between $A_A(C)/q\mathbb{Z}^n$ and $C$. The second one comes from the fact that since $\Lambda \subset \mathbb{Z}^n$, it follows that $\det \Lambda \in \mathbb{Z}$. Taking $q = \det \Lambda$ and $M$ a generator matrix for $\Lambda$, the linear system $Mx = qz$ has integer solution for any $z \in \mathbb{Z}^n$, and therefore $q\mathbb{Z}^n \subset \Lambda$ ($\Lambda$ is $q$-ary).

The last property shows that when dealing with $q$-ary lattices we are indeed considering any integer lattice.

3 The $p$-Lee metric

Instead of the usual Hamming metric for codes and Euclidean metric for lattices we consider here the $l_p$ norm for $A_A(C) \subset \mathbb{Z}^n$ and the induced $p$-Lee metric for the associated code $C$. We show later (Section 4) how this metric naturally arises from the study of lattices considering the $l_p$ norm. We can regard the search for the closest lattice point in $A_A(C)$ as a decoding problem in $C$.

Let $x = (x_1, \ldots, x_n)^t, y = (y_1, \ldots, y_n)^t \in \mathbb{R}^n$. The usual $l_p$ distance (denoted by $d_p$) between these vectors is defined as

$$d_p(x, y) := \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p} \quad \text{if } 1 \leq p < \infty$$

(2)

and

$$d_\infty(x, y) := \max\{|x_i - y_i|; \; i = 1, \ldots, n\}.$$  

(3)

The Lee metric, introduced in [7], can be viewed as the distance in $\mathbb{Z}_q^n$ induced by $d_1$-metric in $\mathbb{Z}^n$. For two elements $\overline{x}, \overline{y} \in \mathbb{Z}_q$

$$d_{\text{Lee}}(\overline{x}, \overline{y}) = \min\{|(\overline{x} - \overline{y}) \mod q\}, (\overline{y} - \overline{x}) \mod q\},$$

(4)

whereas for two vectors $\overline{x}, \overline{y} \in \mathbb{Z}_q^n$, we have

$$d_{1,\text{Lee}}(\overline{x}, \overline{y}) = \sum_{i=1}^n \min\{|(\overline{x}_i - \overline{y}_i) \mod q\}, (\overline{y}_i - \overline{x}_i) \mod q\}. $$

(5)

A generalization of this metric is proposed in the next Proposition 3. We will denote by $\lceil \rceil$ the rounding to the nearest integer.
Therefore we can assert
\[
    d(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^{n} (d_{\text{Lee}}(\mathbf{x}_i, \mathbf{y}_i))^p \right)^{1/p},
\]
where \(d_{\text{Lee}}(\mathbf{x}, \mathbf{y})\) is given by Equation (4). We will denote this distance as \(d_{p,\text{Lee}}(\mathbf{x}, \mathbf{y}) = d(\mathbf{x}, \mathbf{y})\).

Proof Let \(\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_q^n, 0 \leq x_i, y_i < q\). The induced metric in \(\mathbb{Z}_q^n\) by the metric \(d_p\) is, by definition, given by
\[
    d(\mathbf{x}, \mathbf{y}) = \inf\{d_p(x^*, y^*); x^* = x + qt, y^* = y + qw; t, w \in \mathbb{Z}^n\}
\]
\[
    = \left( \inf \left\{ \sum_{i=1}^{n} |x_i - y_i - q(x_i - y_i)|^p; t, w \in \mathbb{Z}^n \right\} \right)^{1/p}
\]
\[
    = \left( \sum_{i=1}^{n} |x_i - y_i - q(x_i - y_i)|^p \right)^{1/p},
\]
where the last equality follows from the fact that \(\sum_{i=1}^{n} |x_i - y_i - qs_i|^p\) is minimum when \(s_i = \left\lceil \frac{x_i - y_i}{q} \right\rceil\), since the summation terms are independent.

Let \(\alpha_i = \left\lceil \frac{x_i - y_i}{q} \right\rceil\) for \(i = 1, \ldots, n\). Since \(0 \leq |x_i - y_i| \leq q\), it follows that \(-1 \leq \frac{x_i - y_i}{q} \leq 1\) and \(\alpha_i \in \{-1, 0, 1\}\).
- If \(\alpha_i = 0\) for some \(i\), then \(-q/2 \leq x_i - y_i \leq q/2\) and this implies \(\min\{|x_i - y_i|, q - |x_i - y_i|\} = |x_i - y_i|\).
- If \(\alpha_i = 1\) for some \(i\), then \(q/2 \leq x_i - y_i \leq q\) and then \(\min\{|x_i - y_i|, q - |x_i - y_i|\} = q - |x_i - y_i|\) and \(|x_i - y_i| = x_i - y_i|\).
- If \(\alpha_i = -1\) for some \(i\), then \(-q \leq x_i - y_i < -q/2\) and then \(\min\{|x_i - y_i|, q - |x_i - y_i|\} = q - |x_i - y_i|\) and \(|x_i - y_i| = -x_i - y_i|\).

Therefore we can assert \(d(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^{n} (d_{\text{Lee}}(\mathbf{x}_i, \mathbf{y}_i))^p \right)^{1/p}\). \(\square\)

For \(p = \infty\) a similar result holds.

Proposition 3 For \(\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_q^n, 0 \leq x_i, y_i < q\), the induced metric in \(\mathbb{Z}_q^n\) by the metric \(d_\infty\) is given by
\[
    d_{\infty,\text{Lee}}(\mathbf{x}, \mathbf{y}) := \max\{|x_i - y_i|; i = 1, \ldots, n\}.
\]

We denote by \(d_{p,\text{Lee}}(C)\) the minimum distance of a code \(C \subseteq \mathbb{Z}_q^n\), and by \(B_{p,\text{Lee}}(R, \mathbf{x})\) the closed ball of radius \(R\) in \(\mathbb{Z}_q^n\) centered at \(\mathbf{x}\). The concept of error correction capacity (or packing radius) \(R\) of a code \(C\) is the greatest \(R\)
such that the balls of radius $R$, $B_{p,Lee}(R, \mathbf{x})$ centered at the distinct points of $C$ are disjoint.

In the Lee metric (i.e., for $p = 1$), the error capacity correction of a code $C$ is given by the well-known formula $t = \lceil (d_{1,Lee}(C) - 1)/2 \rceil$. For $p > 1$, this is not true. For example, take the 13-ary code $C = \langle (1, 5) \rangle = \{j(1, 5) \mod 13 : j = 0, \ldots, 12\}$. For $p = 2$, we have $\lceil (d_{Lee,2}(C) - 1)/2 \rceil = 1$, but the spheres of radius 2 centered at codewords are disjoint, and therefore the packing radius is strictly greater than $\lceil (d_{Lee,2}(C) - 1)/2 \rceil$. The same will occur for distances $d_{p,Lee}$, $p > 2$. If $C$ is a $q$-ary code, the minimum distance $\mu = \min\{d_p(\mathbf{x}, \mathbf{0}), \mathbf{0} \neq \mathbf{x} \in A_A(C)\}$ of $A_A(C)$ is related to the minimum distance $d_{p,Lee}(C)$ of $C$, $\mu = \min\{q, d_{p,Lee}(C)\}$ [10].

4 Decoding Construction A lattices

For $q$-ary lattices constructed from $q$-ary codes, we have shown in [1] that decoding a $q$-ary code $C \subseteq \mathbb{Z}_q^n$ considering the Lee metric corresponds to decoding the associated $q$-ary lattice $A_A(C) \subseteq \mathbb{R}^n$ considering the sum metric $l_1$. In this section, we get the same kind of relation between codes in the $p$-Lee metric and lattices in the $l_p$ metric.

In order to simplify the notation, $\mathbf{r}$ is a codeword of $C$, whereas $\mathbf{x}$, is a point of $A_A(C)$. Due to the isomorphism $A_A(C)/q\mathbb{Z}^n \simeq C$, we will not distinguish the elements of $A_A(C)/q\mathbb{Z}^n \subseteq \mathbb{R}^n/q\mathbb{Z}^n$ from the codewords of $C$.

**Proposition 4** Let $A_A(C)$ be a $q$-ary lattice and $\mathbf{r} = (r_1, \ldots, r_n)^t \in \mathbb{R}^n$ a received vector. Let $\mathbf{r} = (r_1, \ldots, r_n)^t \in \mathbb{R}^n/q\mathbb{Z}^n$ and $\mathbf{c} = (c_1, \ldots, c_n)^t$, $0 \leq c_i < q$, a nearest codeword to $\mathbf{r}$ considering the $p$-Lee metric in $\mathbb{R}^n/q\mathbb{Z}^n$. An element $\mathbf{z} \in A_A(C)$ which is closest to $\mathbf{r}$ considering the $d_p$ metric in $\mathbb{R}^n$ is $\mathbf{z} = (z_1, \ldots, z_n)^t$, where $z_i = c_i + qw_i$ and $w_i = \left\lceil \frac{r_i - x_i}{q} \right\rceil$, for each $i = 1, \ldots, n$.

**Proof** Using the same idea of the proof of Proposition 3 we show that if $\mathbf{c} \in C$, $\mathbf{z} = \mathbf{x} + qw$ where $w_i = \left\lceil \frac{r_i - x_i}{q} \right\rceil$, then $d_p(\mathbf{r}, \mathbf{z}) = d_{p,Lee}(\mathbf{r}, \mathbf{z})$. By hypothesis, $\mathbf{c} \in C$ satisfies $d_{p,Lee}(\mathbf{r}, \mathbf{c}) = \min\{d_{p,Lee}(\mathbf{r}, \mathbf{r}), \mathbf{r} \in C\}$. For $w_i = \left\lceil \frac{z_i - x_i}{q} \right\rceil$, it follows that $d_p(\mathbf{r}, \mathbf{c} + qw) = d_{p,Lee}(\mathbf{r}, \mathbf{c}) \leq \min\{d_p(\mathbf{r}, \mathbf{x} + qt), \mathbf{x} \in C, t \in \mathbb{Z}^n\}$, therefore $\mathbf{c} + qw$ is the element of $A_A(C)$ that minimizes the distance $d_p$ to $\mathbf{r}$.

5 Perfect codes in the $p$-Lee metric

Given a metric $d$, a code $C \subseteq \mathbb{Z}_q^n$ such that for any $\mathbf{r} \in \mathbb{Z}_q^n$ there is only one codeword $\mathbf{c} \in C$ such that $d(\mathbf{r}, \mathbf{c}) \leq R$ is called a perfect code or a $R$-perfect code. The characterization of perfect codes is completely solved in the case of the Hamming metric and an open problem in the Lee (1-Lee) metric, in which case the famous Golomb-Welch [5] conjecture attempts to provide an answer.
Let $\mu_p(n,R)$ denote the number of points in $\mathbb{Z}_q^n$ inside a ball in the $p$-Lee metric of radius $R$ centered at the origin. Let $C \subseteq \mathbb{Z}_q^n$ be a code. The following is straightforward:

**Proposition 5 (Sphere-Packing Bound)** If the minimum distance of a code $C \subseteq \mathbb{Z}_q^n$ is $2R + 1$, then $|C| \mu_p(n,R) \leq q^n$.

The proposition holds for any metric $d$ which is invariant under translation and comes from the fact that $\bigcup_{\mathbf{x} \in C} B_{\text{Lee},p}(\mathbf{x},R) \subseteq \mathbb{Z}_q^n$. 

Codes such that the inclusion above is an equality are perfect codes. The trivial perfect codes are $C = \mathbb{Z}_q^n$ and $C = \{0\}$. For $p = 1$ and $p = \infty$, there are closed forms for $\mu_p(n,R)$, where $2R + 1 \leq q$ namely

$$\mu_1(n,R) = \min \{n,R\} \sum_{i=0}^{\min \{n,R\}} 2^i \binom{n}{i} \binom{R}{i}$$

and

$$\mu_\infty(n,R) = (2R + 1)^n,$$

however for the general case there is no such formulas.

Considering the Lee metric ($p = 1$), it is easy to exhibit examples of perfect codes for $n = 2$ and any $R$ and the so-called Golomb-Welch conjecture proposes that for $n > 2$ there are only perfect codes for $R = 1$ [5]. Related results concerning the $p$-Lee metric are described next.

**Proposition 6** For $1 \leq p < \infty$, there are perfect codes in $\mathbb{Z}_q^n$ in the $p$-Lee metric for $R = 1$ and $q = 2n + 1$.

**Proof** The case $p = 1$ is a classic result and was proved in [5, Thm. 3]. For $1 < p < \infty$ the equation $|x_1|^p + \ldots + |x_n|^p \leq 1$ has exactly $2n + 1$ integer solutions, namely $x_i = \pm 1$ and $x_j = 0$ for all $j \neq i$ and $x_i = 0$ for all $i$. Therefore $\mu_p(n,1) = 2n + 1 = \mu_1(n,1)$. Since there is a perfect code $C \subseteq \mathbb{Z}_q^n$ in the 1-Lee metric satisfying the conditions of the proposition, it follows that this code is also a perfect code in the $p$-Lee metric for any $1 < p < \infty$ since $|C| \mu_p(n,1) = |C| \mu_1(n,1) = q^n$. \[\square\]

Perfect codes in the case $p = \infty$ can be fully characterized through the following proposition:

**Proposition 7** There are non-trivial perfect codes $C \subseteq \mathbb{Z}_q^n$ in the $\infty$-Lee metric iff $q = bm$ with $b > 1$ an odd integer and $m > 1$ an integer.

**Proof** A code $C \subseteq \mathbb{Z}_q^n$ with minimum distance $2R + 1$ is perfect with respect to the distance $d_{\infty,\text{Lee}}$ iff

$$|C|(2R + 1)^n = q^n.$$
Necessary condition: For the above condition (9), if there exists a perfect code $C$, so

$$|C| = \left( \frac{q}{2R + 1} \right)^n.$$ 

Hence $2R + 1$ has to divide $q$. Excluding the trivial codes, $q$ must have an odd factor greater than 1, showing that $q = 2^a$ is impossible. If $q$ is prime, since $(2R + 1)|q$ it follows that $2R + 1 = q$, which gives a trivial perfect code. This shows that there is no perfect code neither for $q$ prime nor for $q$ a power of two, proving the necessary condition.

Sufficient condition: Let $q = bm$ with $b > 1$ an odd integer and $m > 1$ an integer. Taking the code $C$ generated by the vectors

$$\{(b, 0, \ldots, 0), (0, b, \ldots, 0), \ldots, (0, \ldots, 0, b)\} \subseteq \mathbb{Z}_q^n$$

we have $|C| = m^n$. In fact, if $\bar{t} \in \mathbb{Z}_q$ and $\bar{t} = \bar{m}m + \bar{r}$ with $\bar{m} \leq \bar{r} < \bar{m}$ then $\bar{t}(0, \ldots, 0, b, \ldots, 0) = \bar{r}(0, \ldots, 0, b, \ldots, 0)$. For this code the minimum distance

$$\mu = \min\{d_{\infty, \text{Lee}}(\bar{x}, \bar{y}); \bar{x}, \bar{y} \in C, \bar{x} \neq \bar{y}\} = b.$$ 

Therefore $R = (b - 1)/2$. Since $\mu_{\infty}(n, R) = (2R + 1)^n = b^n$, it follows that $|C|^{\mu_{\infty}(n, R)} = m^n b^n = q^n$, $1 < |C| < q^n$ and this code is perfect and non-trivial.

Proposition 8 Let $n \geq 2$ and $1 < p < \infty$. There is $\bar{R}_{n,p}$ such that for $R > \bar{R}_{n,p}$ there is no code $C$ in the $p$-Lee metric that reaches the Sphere-Packing bound.

This shows that for fixed $n$ and $p$ the set of possible minimum distances such that there exists perfect codes in the $p$-Lee metric is finite. We close this paper with an extension to the Golomb-Welch conjecture for the $p$-Lee metrics ($p \neq \infty$):

Conjecture 1 There are no perfect codes in the $p$-Lee metric, $p \neq \infty$, for $n > 2$ and $R > 1$.

References
On Low Weight Codewords of Generalized Affine and Projective Reed-Muller Codes (Extended abstract)

Stéphane Ballet · Robert Rolland

Received: date / Accepted: date

Abstract We propose new results on low weight codewords of affine and projective generalized Reed-Muller codes. In the affine case we give some results on codewords that cannot reach the second weight also called the next to minimal distance. In the projective case the second distance of generalized Reed-Muller codes is estimated, namely a lower bound and an upper bound of this weight are given.

Keywords code · codeword · finite field · generalized Reed-Muller code · homogeneous polynomial · hyperplane · hypersurface · minimal distance · next-to-minimal distance · polynomial · projective Reed-Muller code · second weight · weight

Mathematics Subject Classification (2010) 94B27 · 94B65 · 11G25 · 11T71

1 Introduction - Notations

This paper proposes a study on low weight codewords of generalized Reed-Muller codes and projective generalized Reed-Muller codes of degree $d$, defined over a finite field $\mathbb{F}_q$, called respectively GRM codes and PGRM codes.

For GRM codes, we give some results concerning the next to minimal weight codewords. These codewords are known when $1 \leq d \leq \frac{q}{2}$ (cf. [10], [21]). For other values of $d$ we prove that an irreducible, non-absolutely irreducible polynomial cannot reach the second weight. For $d < q - 1$ we improve the previous result. More precisely we show that a polynomial having a factor of degree $d \geq 2$ which is irreducible, non-absolutely irreducible, cannot reach the second weight.

For PGRM codes, we determine an upper bound and a lower bound for the second weight of a PGRM code.

Determining the low weights of the Reed-Muller codes as well as the low weight codewords are interesting questions related to various fields. Of course, from the point of view

Stéphane Ballet
Aix-Marseille University, ERISCS and IML case 930, F13288 Marseille cedex 9, France
E-mail: stephane.ballet@univ-amu.fr

Robert Rolland
Aix-Marseille University, ERISCS and IML case 930, F13288 Marseille cedex 9 France
E-mail: robert.rolland@acrypta.fr
of coding theory, knowing something on the weight distribution of a code, and especially on the low weights is a valuable information. From the point of view of algebraic geometry the problem is also related to the computation of the number of rational points of hypersurfaces and in particular hypersurfaces that are arrangements of hyperplanes. By means of incidence matrices, Reed-Muller codes are related to finite geometry codes (see [1, 5.3 and 5.4]). From this point of view, the codewords have a geometrical interpretation and can benefit from the numerous results in this area. Consequently there is a wide variety of concepts that may be involved.

1.1 Polynomials and homogeneous polynomials

Let \( \mathbb{F}_q \) be the finite field with \( q \) elements and \( n \geq 1 \) an integer. We denote respectively by \( \mathcal{A}(q) \) and \( \mathcal{P}(q) \) the affine space and the projective space of dimension \( n \) over \( \mathbb{F}_q \).

Let \( \mathbb{F}_q[X_1, X_2, \ldots, X_n] \) be the algebra of polynomials in \( n \) variables over \( \mathbb{F}_q \). If \( f \) is in \( \mathbb{F}_q[X_1, X_2, \ldots, X_n] \) we denote by \( \deg(f) \) its total degree and by \( \deg_i(f) \) its partial degree with respect to the variable \( X_i \).

Denote by \( \mathcal{F}(q, n) \) the space of functions from \( \mathbb{F}_q^n \) into \( \mathbb{F}_q \). Any function in \( \mathcal{F}(q, n) \) can be represented by a unique reduced polynomial \( f \), namely such that for any variable \( X_i \) the following holds:

\[
\deg_X(f) \leq q - 1.
\]

We denote by \( \mathcal{P}(q, n) \) the set of reduced polynomials in \( n \) variables over \( \mathbb{F}_q \).

Let \( d \) be a positive integer. We denote by \( \mathcal{P}(q, n, d) \) the set of reduced polynomials \( P \) such that \( \deg(P) \leq d \). Remark that if \( d \geq n(q-1) \) the set \( \mathcal{P}(q, n, d) \) is the whole set \( \mathcal{P}(q, n) \).

Let \( \mathcal{H}(q, n+1, d) \) the space of homogeneous polynomials in \( n+1 \) variables over \( \mathbb{F}_q \) with total degree \( d \). The decomposition

\[
\mathbb{F}_q[X_0, X_1, X_2, \ldots, X_n] = \bigoplus_{d \geq 0} \mathcal{H}(q, n+1, d)
\]

provides \( \mathbb{F}_q[X_0, X_1, X_2, \ldots, X_n] \) with a graded algebra structure.

1.2 Generalized Reed-Muller codes

Let \( d \) be an integer such that \( 1 \leq d < n(q-1) \). The generalized Reed-Muller code (GRM code) of order \( d \) over \( \mathbb{F}_q \) is the following subspace of \( \mathbb{F}_q^{(q^n)} \):

\[
\text{RM}_d(q, n) = \left\{ (f(X))_{X \in \mathbb{F}_q} \mid f \in \mathbb{F}_q[X_1, \ldots, X_n] \text{ and } \deg(f) \leq d \right\}.
\]

It may be remarked that the polynomials \( f \) determining this code are viewed as polynomial functions. Hence each codeword is associated with a unique reduced polynomial in \( \mathcal{P}(q, n, d) \).

Let us denote by \( Z_d(f) \) the set of zeros of \( f \) (where the index \( d \) stands for "affine"). From a geometrical point of view \( Z_d(f) \) is an affine algebraic hypersurface in \( \mathbb{F}_q^n \) and the number of points \( N_d(f) = \#Z_d(f) \) of this hypersurface (the number of zeros of \( f \)) is connected to the weight \( W_d(f) \) of the associated codeword by the following formula:

\[
W_d(f) = q^n - N_d(f).
\]
The code $\text{RM}_q(d,n)$ has the following parameters (cf. [12], [3, p. 72]) (where the index $a$ stands for “affine code”):
1. length $m_a(q,n,d) = q^n$,
2. dimension

$$k_a(q,n,d) = \sum_{i=0}^{d} \sum_{j=0}^{n} (-1)^{i} \binom{n}{j} \left( \frac{t-jq+n-1}{t-jq} \right),$$
3. minimum distance $W_a^{(1)}(q,n,d) = (q-b)q^{n-a-1}$, where $a$ and $b$ are the quotient and the remainder in the Euclidean division of $d$ by $q-1$, namely $d = a(q-1) + b$ and $0 \leq b < q-1$.

We denote by $N_a^{(1)}(q,n,d)$ the maximum number of zeros for a non-null polynomial function of degree $\leq d$ where $1 \leq d < n(q-1)$, namely

$$N_a^{(1)}(q,n,d) = q^n - W_a^{(1)}(q,n,d) = q^n - (q-b)q^{n-a-1}.$$

The minimum distance of $\text{RM}_q(d,n)$ was given by T. Kasami, S. Lin, W. Peterson in [12]. The words reaching this bound were characterized by P. Delsarte, J. Goethals and F. MacWilliams in [8].

1.3 Projective generalized Reed-Muller codes

The case of projective codes is a bit different, because homogeneous polynomials do not define in a natural way functions on the projective space. Let $d$ be an integer such that $1 \leq d \leq n(q-1)$. The projective generalized Reed-Muller code of order $d$ (PGRM code) was introduced by G. Lachaud in [14]. Let $S$ a subset of $\mathbb{F}_q^{n+1}$ constituted by one point on each punctured vector line of $\mathbb{F}_q^{n+1}$. Remark that any point of the projective space $\mathbb{P}^n(q)$ has a unique coordinate representation by an element of $S$. The projective Reed-Muller code $\text{PRM}_a(n,d)$ of order $d$ over $\mathbb{P}^n(q)$ is constituted by the words $(f(X))_{X \in S}$ where $f \in \mathcal{H}(q,n+1,d)$ and the null word:

$$\text{PRM}_a(n,d) = \{ (f(X))_{X \in S} \mid f \in \mathcal{H}(q,n+1,d) \} \cup \{ 0, \ldots, 0 \}.$$ 

This code is dependent on the set $S$ chosen to represent the points of $\mathbb{P}^n(q)$. But the main parameters are independent of this choice. Following [14] we can choose

$$S = \cup_{i=0}^{n} S_i,$$

where $S_i = \{ 0, \ldots, 0, 1, x_{i+1}, \ldots, x_n \mid x_k \in \mathbb{F}_q \}$. Subsequently, we shall adopt this value of $S$ to define the code $\text{PRM}_a(n,d)$.

For a homogeneous polynomial $f$ let us denote by $Z_h(f)$ the set of zeros of $f$ in the projective space $\mathbb{P}^n(q)$ (where the index $h$ stands for “projective”). From a geometrical point of view, an element $f \in \mathcal{H}(q,n+1,d)$ defines a projective hypersurface $Z_h(f)$ in the projective space $\mathbb{P}^n(q)$. The number $N_h(f) = \#Z_h(f)$ of points of this projective hypersurface is connected to the weight $W_h(f)$ of the corresponding codeword by the following relation:

$$W_h(f) = q^{n+1} - 1 - N_h(f).$$

The parameters of $\text{PRM}_a(n,d)$ are the following (cf. [23]) (where the index $h$ stands for “projective code”):
1. length $m_h(q, n, d) = \frac{q^{d+1} - 1}{q-1}$.
2. dimension

\[ k_l(q, n, d) = \sum_{j=0}^{d} \frac{q + 1}{j} \left( \begin{array}{c} n + 1 \vspace{1pt} \rule{0pt}{0pt} \end{array} j \right) \times \left( \begin{array}{c} t - jq + n \\ t - jq \end{array} \right), \]

3. minimum distance: $W_b^{(1)}(q, n, d) = (q - b)q^{n-a-1}$ where $a$ and $b$ are the quotient and the remainder in the Euclidean division of $d - 1$ by $q - 1$, namely $d - 1 = a(q - 1) + b$ and $0 \leq b < q - 1$.

We denote by $N_h^{(1)}(q, n, d)$ the maximum number of zeros for a non-null homogeneous polynomial function of degree $d$ where $1 \leq d \leq n(q - 1)$, namely

\[ N_h^{(1)}(q, n, d) = \frac{q^{n+1} - 1}{q - 1} - W_b^{(1)}(q, n, d) = \frac{q^{n+1} - 1}{q - 1} - (q - b)q^{n-a-1}. \]

1.4 Minimal distance and corresponding codewords

1.4.1 The affine case: GRM codes

For the affine case recall that we write the degree $d$ in the following form:

\[ d = a(q - 1) + b \quad \text{with } 0 \leq b < q - 1. \]

The minimum distance of a GRM code was given by T. Kasami, S. Lin, W. Peterson in [12]. The words reaching this bound (i.e. the polynomials reaching the maximal number of zeros) were characterized by P. Delsarte, J. Goethals and F. MacWilliams in [8]. Such a polynomial will be called a maximal polynomial and the associated hypersurface is called a maximal hypersurface. The corresponding weight is the minimal weight.

1.4.2 The projective case: PGRM codes

Let us denote respectively by $W_b^{(1)}(q, n, d)$ and $W_b^{(2)}(q, n, d)$ the first and second weight of the projective Reed-Muller code.

In order to describe the minimal distance for the projective case, write $d - 1 = a(q - 1) + b$ with $0 \leq b < q - 1$. The minimum distance of a PGRM code was given by J.-P. Serre for $d \leq q$ (cf. [22]), and by A. Sorensen in [23] for the general case. The polynomials reaching the maximal number of zeros (or defining the minimum weighted codewords) are given by J.-P. Serre for $d \leq q$ (cf. [22]) and by the last author (cf. [19]) for the general case.

2 The second weight in the affine case

2.1 what is known

Let us denote by $W_b^{(2)}(q, n, d)$ the second weight of the GRM code $RM_q(d, n)$, namely the weight which is just above the minimum distance. Several simple cases can be easily described. If $d = 1$, we know that the code has only three weights: 0, the minimum distance $W_a^{(1)}(q, n, 1) = q^n - q^{n-1}$ and the second weight $W_a^{(2)}(q, n, 1) = q^n$. For $d = 2$ and
For all cases except when \( d = 2 \), and any \( q \) (including \( q = 2 \)) the weight distribution was given by R. McEliece in [18]. For \( q = 2 \), for any \( n \) and any \( d \), the weight distribution is known in the range \([W_a^{(1)}(2, n, d), 2.5W_a^{(1)}(2, n, d)]\) by a result of Kasami, Tokura, Azumi [13]. In particular, the second weight is \( W_a^{(2)}(2, n, d) = 3 \times 2^{n-d-1} \) if \( 1 < d < n-1 \) and \( W_a^{(2)}(2, n, d) = 2^{n-d+1} \) if \( d = n-1 \) or \( d = 1 \). For \( d \geq n(q-1) \) the code \( \text{RM}_q(d, n) \) is trivial, namely it is the whole \( \mathcal{F}(q, d, n) \), hence any integer \( 0 \leq t \leq q^d \) is a weight. Let us remark also that if \( q = p \) is a prime, GRM codes (and also PGRM codes) are the finite geometry codes, and in this case the next to minimal distance is known as well as the geometrical nature of the corresponding codewords.

The general problem of the second weight was tackled by D. Erickson in his thesis [10, 1974] and was partly solved. Unfortunately this very good piece of work was not published and remained virtually unknown. Meanwhile several authors became interested in the problem. The second weight was first studied by J.-P. Cherdieu and R. Rolland in [7] who proved that when \( q > 2 \) is fixed, for \( d < q \) sufficiently small the second weight is

\[
W_a^{(2)}(q, n, d) = q^d - dq^{d-1} + (d - 1)q^{d-2}.
\]

Their result was improved by A. Sboui in [21], who proved the formula for \( d \leq q/2 \). The methods in [7] and [21] are of a geometric nature by means of which the codewords reaching this weight were determined. These codewords are hyperplane arrangements. Then O. Geil in [11], using Gröbner basis methods, proved the formula for \( d < q \) and solved the problem for \( n = 2 \). This case is particularly important as we shall see later. Finally, the last author in [20], using a mix of Geil’s method and geometrical considerations found the second weight for all cases except when \( d = a(q-1) + 1 \).

Recently, A. Bruen ([6]) exhumes the work of Erickson and completed the proof, solving the problem of the second weight for Generalized Reed-Muller code. Let us describe more precisely the result of Erickson. First, in order to present his result let us introduce the following notation used in [10]: \( s \) and \( t \) are integers such that

\[
d = s(q-1) + t, \text{ with } 0 < t \leq q - 1.
\]

**Theorem 1** The second weight \( W_a^{(2)}(q, n, d) \) is

\[
W_a^{(2)}(q, n, d) = W_a^{(1)}(q, n, d) + cq^{d-t-2}
\]

where \( W_a^{(1)}(q, n, d) = (q-t)q^{d-t-1} \) is the minimal distance and \( c \) is

\[
c = \begin{cases} 
q & \text{if } s = n - 1 \\
t - 1 & \text{if } s < n - 1 \text{ and } 1 < t \leq \frac{q+1}{2} \\
 & \text{or } s < n - 1 \text{ and } t = q-1 \neq 1 \\
q & \text{if } s = 0 \text{ and } t = 1 \\
q - 1 & \text{if } q < 4, s < n - 2 \text{ and } t = 1 \\
q - 1 & \text{if } q = 3, s = n - 2 \text{ and } t = 1 \\
q & \text{if } q = 2, s = n - 2 \text{ and } t = 1 \\
q & \text{if } q \geq 4, 0 < s \leq n - 2 \text{ and } t = 1 \\
c_t & \text{if } q \geq 4, s \leq n - 2 \text{ and } \frac{q+1}{2} < t
\end{cases}
\]

The number \( c_t \) is such that \( c_t + (q-t)q \) is the second weight for the code \( \text{RM}_q(2, t) \).
Unfortunately the number \(c_t\) is not determined in the work of Erickson. But, it results from the previous theorem that if someone could calculate the second weight for a case where \(e = c_t\), the problem would be fully resolved. Alternatively, Erickson conjectured that \(c_t = t - 1\) and reduced this conjecture to a conjecture on blocking sets [10, Conjecture 4.14 p. 76]. Recently in [6] A. Bruen proved that this conjecture follows from two of his papers [4], [5]. Then the problem is now solved by [10]+[6]. It is also solved by [10]+[11] (the important case \(n = 2\) is completely solved in [11] and this leads to the conclusion as noted above) or by [10]+[20] (the cases not solved in [10] are explicitly resolved in [20]).

Remark 2 The values \(s\) and \(t\) are connected to the values \(a\) and \(b\) of the formula (1) in the following way: \(a = s\) and \(b = t\) unless \(t = q - 1\) and in this case \(a = s + 1\) and \(b = 0\). Then we can also express the second weight with the classical writing for the Euclidean quotient as in [20].

Finally let us remark that we now have several approaches, close to each other, but nevertheless different. The first one [10],[6] is mainly based on combinatorics of finite geometries, the second one [7],[21],[20] is mainly based on geometry and hyperplane arrangements, the third [11],[20] is mainly based on polynomial study by means of commutative algebra and Gröbner basis. All these approaches can be fruitful for the study of similar problems, in particular for the similar codes based on incidence structures, finite geometry and incidence matrices (see [24],[16],[17],[15]).

2.2 New results on the codewords reaching the second weight

The polynomials reaching the second weight are known for \(2d \leq q\) (cf. [10, Theorem 3.13, p. 60],[21]). For the other values of \(d\) the result is not known. However we can say that:

**Theorem 3** If \(f \in \mathbb{R} P(q, n, d)\) is an irreducible polynomial but not absolutely irreducible, in \(n\) variables over \(\mathbb{F}_q\), of degree \(d > 1\) then the weight \(W_a(f)\) of the corresponding codeword in \(\text{RM}_q(n, d)\) is such that \(W_a(f) > W_a^{(2)}(q, n, d)\). Namely such a polynomial cannot reach the next to minimal weight.

**Theorem 4** If \(f \in \mathbb{R} P(q, n, d)\) is a product of two polynomials \(f = gh\) such that

1. \(2 \leq d' = \deg(g) \leq d = \deg(f) < q - 1\);
2. \(g\) is irreducible but not absolutely irreducible;

then \(W_a(f) > W_a^{(2)}(q, n, d)\). Namely such a polynomial cannot reach the next to minimal weight.

The proofs of these two theorems will be given in the full paper and can be found in the preprint [2].

**Remark 5** In any case, among the words reaching the second distance, there are hyperplane configurations. For example the hyperplane configurations given in [20].
3 The second weight in the projective case

In this section we tackle the problem of finding the second weight \( W_h^{(2)}(q, n, d) \) for GPRM codes. Note that if \( q \) is a prime \( p \).

**Lemma 6** Let \( f \) be a homogeneous polynomial in \( n+1 \) variables of total degree \( d \), with coefficients in \( \mathbb{F}_q \), which does not vanish on the whole projective space \( \mathbb{P}^n(q) \). If there exists a projective hyperplane \( H \) such that the affine hypersurface \( (\mathbb{P}^n(q) \setminus H) \cap Z_\Delta(f) \) contains an affine hyperplane of the affine space \( \mathbb{A}^n(q) = \mathbb{P}^n(q) \setminus H \) then the projective hypersurface \( Z_\Delta(f) \) contains a projective hyperplane. In particular if \( f \) restricted to the affine space \( \mathbb{A}^n(q) \) defines a maximal affine hypersurface then \( Z_\Delta(f) \) contains a hyperplane.

**Lemma 7** For \( n \geq 2 \) the following holds
\[
W_h^{(1)}(q, n-1, d) + W_a^{(2)}(q, n, d) \leq W_a^{(2)}(q, n, d-1).
\]

**Proof** Let us introduce the following notations:
\[
d - 1 = s_{d-1}(q - 1) + t_{d-1}, \quad \text{where} \quad 1 \leq t_{d-1} \leq q - 1;
\]
\[
d = s_d(q - 1) + t_d, \quad \text{where} \quad 1 \leq t_d \leq q - 1.
\]
The values \( c(d - 1) \) and \( c(d) \) are the values of the coefficient \( c \) which occurs in Theorem 1, with respect to \( d - 1 \) and \( d \). Then we have
\[
W_h^{(1)}(q, n-1, d) = (q - t_{d-1})q^{n-d+1-1} - 2,
\]
\[
W_a^{(2)}(q, n, d) = (q - t_d)q^{n-d+1} + c(d)q^{n-d+2},
\]
\[
W_a^{(2)}(q, n, d - 1) = (q - t_{d-1})q^{n-d+1} + c(d - 1)q^{n-d+1-2}.
\]
Denote by \( \Delta \) the difference
\[
\Delta = W_a^{(2)}(q, n, d - 1) - \left( W_h^{(1)}(q, n-1, d) + W_a^{(2)}(q, n, d) \right).
\]
- If \( 1 \leq t_{d-1} \leq q - 2 \) then \( q > 2 \), \( t_d = t_{d-1} + 1 \) and \( s_d = s_{d-1} \). In this case let us denote by \( s \) the common value of \( s_d \) and \( s_{d-1} \). Hence
\[
\Delta = q^{n-d-2}(t_{d-1} + c(d - 1) - c(d)).
\]
  - If \( s = n - 1 \) then \( c(d - 1) = c(d) = q \) and \( \Delta > 0 \).
  - If \( s < n - 1 \) and \( 1 < t_{d-1} \leq \frac{n - 1}{2} \) then \( q \geq 4 \) and \( c(d - 1) - c(d) = -1 \). Hence \( \Delta > 0 \).
  - If \( s < n - 1 \) and \( \frac{n - 1}{2} - 1 < t_{d-1} \leq \frac{n - 1}{2} \) then \( q \geq 4 \) and \( c(d - 1) - c(d) = -1 \). Hence \( \Delta > 0 \).
  - If \( s < n - 1 \), \( q \geq 4 \) and \( t_{d-1} = 1 \) then \( c(d - 1) - c(d) = q - t_{d-1} \). Hence \( \Delta > 0 \).
  - If \( s < n - 1 \) and \( q = 3 \) and \( t_{d-1} = 1 \) then \( c(d - 1) - c(d) = 1 \). Hence \( \Delta > 0 \).
- If \( t_{d-1} = q - 1 \) then \( t_d = 1 \) and \( s_d = s_{d-1} + 1 \). Hence
\[
\Delta = q^{n-d-1-3}(c(d-1)q - c(d)) \geq 0.
\]
Theorem 8 Let $W_h^{(2)} (q, n, d)$ be the second weight for a homogeneous polynomial $f$ in $n + 1$ variables ($n \geq 2$) of total degree $d$, with coefficients in $\mathbb{F}_q$, which is not maximal. Let us define $V_h^{(2)} (q, n, d)$ by:

$$V_h^{(2)} (q, n, d) = 2 \text{ if } d > n(q - 1)$$

and

$$V_h^{(2)} (q, n, d) = W_h^{(1)} (q, n - 1, d) + W_d^{(2)} (q, n, d) \text{ if } d \leq n(q - 1). \quad (2)$$

Then the following holds

$$V_h^{(2)} (q, n, d) \leq W_h^{(2)} (q, n, d) \leq W_a^{(2)} (q, n, d - 1).$$

Proof Let us remark first that by Lemma 7

$$V_h^{(2)} (q, n, d) \leq W_a^{(2)} (q, n, d - 1).$$

If $d > n(q - 1)$, as $f$ does not vanish on the whole projective space $\mathbb{P}^n(q)$, and $f$ is not maximal then $N_0(f) \leq \frac{q^{n+1} - 1}{q - 1} - 2$. This bound is attained. Then in this case $W_h^{(2)} (q, n, d) = 2$.

Suppose now that $2 \leq d \leq n(q - 1)$. Let $f$ such that $Z_0(f)$ is not maximal. Suppose first that there is an hyperplane $H$ in $Z_0(f)$. Then we can suppose that

$$f(X_0, X_1, \cdots, X_n) = X_0g(X_0, X_1, \cdots, X_n)$$

where $g$ is an homogeneous polynomial of degree $d - 1$. The function

$$f_1(X_1, \cdots, X_n) = g(1, X_1, \cdots, X_n)$$

defined on the affine space $\mathbb{A}^n(q) = \mathbb{P}^n(q) \setminus H$ is a polynomial function in $n$ variables of total degree $d - 1$. If it was maximum, by [19, Lemma 2.3] the function $f$ would also be maximum.

Then $#Z_0(f_1) \leq q^n - W_a^{(2)} (q, n, d - 1)$. Hence the following holds:

$$#Z_0(f) \leq \frac{q^n - 1}{q - 1} + q^n - W_a^{(2)} (q, n, d - 1),$$

$$#Z_0(f) \leq \frac{q^{n+1} - 1}{q - 1} - W_d^{(2)} (q, n, d - 1),$$

and the equality holds if and only if $f_1$ reaches the second weight on the affine space $\mathbb{A}^n(q)$. This case actually occurs. Hence for such a word, in general we have

$$W_h(f) \geq W_a^{(2)} (q, n, d - 1),$$

and as the equality occurs, the following holds for the second distance:

$$W_h^{(2)} (q, n, d) \leq W_a^{(2)} (q, n, d - 1).$$

Suppose now that there is not any hyperplane in the hypersurface $Z_0(f)$. Let $H$ be a hyperplane and $\mathbb{A}^n(q) = \mathbb{P}^n(q) \setminus H$. Then as $H \cap Z_0(f) \neq H$

$$#(H \cap Z_0(f)) \leq \frac{q^n - 1}{q - 1} - W_h^{(1)} (q, n - 1, d),$$
and by Lemma 6

\[ \# (Z_h(f) \cap A^n(q)) \leq q^n - W^{(2)}_a(q, n, d). \]

Then

\[ \#Z_h(f) \leq \frac{q^n - 1}{q - 1} - W^{(1)}_h(q, n - 1, d) + q^n - W^{(2)}_a(q, n, d) \]
\[ \leq \frac{q^{n+1} - 1}{q - 1} - \left( W^{(1)}_h(q, n - 1, d) + W^{(2)}_a(q, n, d) \right) \]

and consequently

\[ W_h(f) \geq W^{(1)}_h(q, n - 1, d) + W^{(2)}_a(q, n, d). \]

Then, for the second distance the conclusion of the theorem holds.

References

On the decoding of quasi-BCH codes

Morgan Barbier · Clément Pernet · Guillaume Quintin

2012/12/20

Abstract In this paper we investigate the structure of quasi-BCH codes. In the first part of this paper we show that quasi-BCH codes can be derived from Reed-Solomon codes over square matrices extending the known relation about classical BCH and Reed-Solomon codes. This allows us to adapt the Welch-Berlekamp algorithm to quasi-BCH codes. In the second part of this paper we show that quasi-BCH codes can be seen as subcodes of interleaved Reed-Solomon codes over finite fields. This provides another approach for decoding quasi-BCH codes.

Keywords Quasi-cyclic code · quasi-BCH code · BCH code · Reed-Solomon · interleaved code

Mathematics Subject Classification (2000) 94B35 · 94B60

1 Introduction

Many codes with best known minimum distances are quasi-cyclic codes or derived from them [10,14]. This family of codes is therefore very interesting. Quasi-cyclic codes were studied and applied in the context of McEliece’s cryptosystem [3,16] and Niederreiter’s [12,17]. They permit to reduce the size of keys in opposition to Goppa codes. However, since the decoding of random quasi-cyclic codes is difficult, only quasi-cyclic alternant codes were proposed for the latter cryptosystem. The high structure of alternant codes is actually a weakness and two cryptanalysis were proposed in [8,19].
1.1 Our contributions

In this paper we investigate the structure of quasi-BCH codes. In the first part of this paper we show that quasi-BCH codes can be derived from Reed-Solomon codes over square matrices. It is well known that BCH codes can be obtained from Reed-Solomon codes [15, Theorem 2, page 300]. We extend this property to quasi-BCH codes which allows us to adapt the Welch-Berlekamp algorithm [4,9] to quasi-BCH codes.

**Theorem 1** Let $\Gamma \in M_\ell \times \ell (F_q)$ be a primitive $m$-th root of unity and $C$ be an $\ell$-quasi-BCH code of length $m\ell$, designed minimum distance $\delta$ with respect to the primitive $m$-th root $\Gamma$. Then there exists a Reed-Solomon code $R$ over the ring $M_\ell \times \ell (F_q)$ with parameters $[m, m - \delta + 1, \delta]_{M_\ell \times \ell (F_q)}$ and an $F_q$-linear, $F_q$-isometric embedding $\psi : C \to R$.

In the second part we show that quasi-BCH codes can be seen as subcodes of interleaved Reed-Solomon codes.

**Theorem 2** Let $\Gamma \in M_\ell \times \ell (F_q)$ be a primitive $m$-th root of unity and $C$ be an $\ell$-quasi-BCH code of length $m\ell$, designed minimum distance $\delta$ with respect to the primitive $m$-th root $\Gamma$. Then $C$ is an interleaved code of $\ell$ subcodes of Reed-Solomon codes over $F_{q^\ell}$ in the following sense: there exist $\ell$ Reed-Solomon codes $C_1, \ldots, C_\ell$ over $F_q$ and an isometric isomorphism from $C$, equipped with the $\ell$-block distance, to a subcode of the interleaved code with respect to $C_1, \ldots, C_\ell$.

1.2 Related work

In [11,13], $\ell$-quasi-cyclic codes of length $m\ell$ are seen as $R$-submodules of $R^\ell$ for a certain ring $R$. However, in [11], Gröbner bases are used in order to describe polynomial generators of quasi-cyclic codes whereas in [13], the authors decompose quasi-cyclic codes as direct sums of shorter linear codes over various extensions of $F_q$ (when $\gcd(m,q) = 1$). This last work leads to an interesting trace representation of quasi-cyclic codes. In [6], the approach is more analogous to the cyclic case. The authors consider the factorization of $X^m - 1 \in M_\ell(F_q)[X]$ with reversible polynomials in order to construct $\ell$-quasi-cyclic codes canceled by those polynomials and called $\Omega(P)$-codes. This leads to the construction of self-dual codes and codes beating known bounds. But the factorization of univariate polynomials over a matrix ring remains difficult. In [7] the author gives an improved method for particular cases of the latter factorization problem.

2 Prerequisites

2.1 Reed-Solomon codes over rings

We recall some basic definitions of Reed-Solomon codes over rings in this section. We let $A$ be a ring with identity, we denote by $A^\times$ the group of units of $A$ and by $Z(A)$ the center of $A$, the commutative subring of $A$ consisting of all the elements of $A$ which commute with all the other elements of $A$. We denote by $A[X]$ the ring
of polynomials over \( A \) and by \( A[X]_{< k} \) the polynomials over \( A \) of degree at most \( k - 1 \).

**Definition 1** Let \( f = \sum_{i=0}^{d} f_i X^i \in A[X] \) be a polynomial with coefficients in \( A \) and \( a \in A \). We call **left evaluation of \( f \) at \( a \)** the quantity

\[
f(a) := \sum_{i=0}^{d} f_i a^i \in A
\]

and **right evaluation of \( f \) at \( a \)** the quantity

\[
(a)f := \sum_{i=0}^{d} a^i f_i \in A.
\]

**Remark 1** For \( f, g \in A[X] \) and \( a \in A \), we obviously have \( f(a) = (a)f \) whenever \( a \in \mathbb{Z}(A) \), \( (f + g)(a) = f(a) + g(a) \), \( (a)(f + g) = (a)f + (a)g \). If \( a \) commutes with all the coefficients of \( g \) we also have \((fg)(a) = f(a)g(a)\) and \((a)(gf) = (a)g(a)f\).

**Definition 2** Let \( 0 < k \leq m \) be two integers. Let \( x = (x_1, \ldots, x_m) \) and \( v = (v_1, \ldots, v_m) \) be two vectors of \( A^m \) such that \( x_i - x_j \in A^k \) and \( x_i x_j = x_j x_i \) for all \( i \neq j \) and \( v_i \in A^k \) for all \( i \).

The left submodule of \( A^m \) generated by the vectors

\[
(f(x_1) \cdot v_1, \ldots, f(x_m) \cdot v_m) \in A^m \text{ with } f \in A[X]_{< k}
\]

is called a **left generalized Reed-Solomon code (LGRS)** over \( A \) with parameters \([v, x, k]_A \) or \([n, k] \) or \([x, k]_A \) if there is no confusion on \( m, x \) and \( v \).

The right submodule of \( A^m \) generated by the vectors

\[
(v_1 \cdot (x_1)f, \ldots, v_m \cdot (x_m)f) \in A^m \text{ with } f \in A[X]_{< k}
\]

is called a **right generalized Reed-Solomon code (RGRS)** over \( A \) with parameters \([v, x, k]_A \) or \([m, k] \) \([x, k]_A \) if there is no confusion on \( m, x \) and \( v \). The vector \( x \) is called the **support**. If \( v = (1, \ldots, 1) \), the codes constructed above are called left Reed-Solomon (LRS) and right Reed-Solomon (RRS) codes.

**Definition 3** Let \( x = (x_1, \ldots, x_m) \in A^m \). We call the **Hamming weight of \( x \)** the number of nonzero coordinates.

\[
w(x) := w(x_1, \ldots, x_m) = |\{i : x_i \neq 0\}|.
\]

Let \( y = (y_1, \ldots, y_m) \in A^m \). The **Hamming distance between \( x \) and \( y \)** is

\[
d(x, y) = w(x - y) = |\{i : x_i \neq y_i\}|.
\]

The **minimum distance** of any subset \( S \subseteq A^m \) is defined as

\[
\min \{d(x, y) : x, y \in S \text{ and } x \neq y\}.
\]

**Proposition 1** A LGRS (resp. RGRS) code is a free left (resp. right) submodule of \( A^m \). A LGRS (resp. RGRS) code with parameters \([m, k] \) has minimum distance \( m - k + 1 \).

**Proof** It suffices to see that the maps

\[
A^m \longrightarrow A^m
\]

\[
(a_1, \ldots, a_m) \longmapsto (a_1 v_1, \ldots, a_m v_m)
\]

\[
(a_1, \ldots, a_m) \longmapsto (v_1 a_1, \ldots, v_m a_m)
\]

are respectively left and right isometric automorphisms of \( A^m \). \( \square \)
2.2 Quasi-cyclic and quasi-BCH codes

Quasi-cyclic codes form an important family of codes defined as follow.

**Definition 4** Let \( T : \mathbb{F}_q^n \to \mathbb{F}_q^n \) to be the left cyclic shift defined by
\[
T(c_1, c_2, \ldots, c_n) = (c_2, \ldots, c_n, c_1).
\]
We call any subset of \( \mathbb{F}_q^n \), invariant by \( T^\ell \), an \( \ell \)-quasi-cyclic code over \( \mathbb{F}_q \) of length \( n \). If the context is clear we will simply say \( \ell \)-quasi-cyclic code.

We will focus in this paper on quasi-BCH codes which form a subfamily of quasi-cyclic codes. They can be seen as a generalization of BCH codes in the context of quasi-cyclic codes.

**Definition 5** We say that \( a \in A \) is a primitive \( m \)-th root of unity if \( a^m = 1 \) and \( \forall 0 \leq i < m, (a^i - 1) \in A^\times \).

**Proposition 2** There exists a primitive \( q^{s\ell} - 1 \)-th root of unity in \( M_{\ell}(\mathbb{F}_{qs}) \).

**Proof** The proof can be found in [2, Proposition 16, page 911]. \( \Box \)

**Definition 6** Let \( \Gamma \) be a primitive \( m \)-th root of unity in \( M_{\ell}(\mathbb{F}_{qs}) \) and \( \delta \leq m \). We define the \( \ell \)-quasi-BCH code of length \( m\ell \), with respect to \( \Gamma \), with designed minimum distance \( \delta \), over \( \mathbb{F}_q \) by
\[
Q_{-}{\text{BCH}}_q(m, \ell, \delta, \Gamma) := \left\{ (c_1, \ldots, c_m) \in (\mathbb{F}_q^\ell)^m : \sum_{j=0}^{m-1} (\Gamma^i)^j(c_{j+1})^T = 0 \text{ for } i = 1, \ldots, \delta - 1 \right\}.
\]
Note that \( Q_{-}{\text{BCH}}_q(m, \ell, \delta, \Gamma) \) is an \( \ell \)-quasi-cyclic code.

**Definition 7** The \( \ell \)-block weight of \( (x_{11}, \ldots, x_{1\ell}, \ldots, x_{m1}, \ldots, x_{m\ell}) \in \mathbb{F}_q^{m\ell} \) is defined to be
\[
\text{Block-w}_\ell(x) := |\{ i : (x_{i1}, \ldots, x_{i\ell}) \neq 0 \}|.
\]
The \( \ell \)-block distance between \( x, y \in \mathbb{F}_q^{m\ell} \) is defined to be \( \text{Block-w}_\ell(x - y) \).

3 Reed-Solomon codes and quasi-BCH codes

3.1 The relation between quasi-BCH and Reed-Solomon codes

We show in this section that under certain assumptions on the support of Reed-Solomon codes, the dual of a LRS code is a RRS code. From this fact we show that quasi-BCH can be constructed from Reed-Solomon codes over square matrices rings. In this Subsection we let \( A \) designate a finite ring with identity.

**Definition 8** Let \( x = (x_1, \ldots, x_m) \) and \( y = (y_1, \ldots, y_m) \) be two vectors of \( A^m \). The inner product is defined as
\[
\langle x, y \rangle := \sum_{i=0}^{m} x_i y_i.
\]
Remark 2 Let $S$ be a subset of $A^m$. Then the set $\{x \in A^m : \forall s \in S, \langle s, x \rangle = 0\}$ denoted by $S^\perp$ is called the right dual of $S$ and is a right submodule of $A^m$. Similarly, let $S$ be a subset of $A^m$. Then the set $\{x \in A^m : \forall s \in S, \langle x, s \rangle = 0\}$ denoted by $^\perp S$ is called the left dual of $S$ and is a left submodule of $A^m$. Note that for all $x, y \in A^m$ and $\mu \in A$ we have $\langle x, y \rangle = \langle \mu x, y \rangle$ and $\langle x, y \rangle \mu = \langle x, y \mu \rangle$.

Remark 3 Let $x = (1, \gamma, \gamma^2, \ldots, \gamma^m) \in A^m$ where $\gamma$ is a primitive $m$-th root of unity. Then a RRS or LRS code whose support is $x$ is cyclic.

Proposition 3 Let $\gamma \in A$ be a primitive $m$-th root of unity. Let $x = (1, \gamma, \gamma^2, \ldots, \gamma^m) \in A^m$. Then the right (resp. left) dual of the LGRS (resp. RGRS) code with parameters $[x, x, k]_A$ is the RRS (resp. LRS) code with parameters $[x, m-k]_A$.

Proof We denote respectively by $\mathcal{L}$ and $\mathcal{R}$ the left generalized Reed-Solomon code with parameters $[x, x, k]_A$ and the right Reed-Solomon code with parameters $[x, m-k]_A$.

First note that $\mathcal{L}$ is generated by the vectors $(1, \gamma^i, \gamma^{2i}, \ldots, \gamma^{(m-1)i})$ for $i = 1, \ldots, k$ and that $\mathcal{R}$ is generated by the vectors $(1, \gamma^i, \gamma^{2i}, \ldots, \gamma^{(m-1)i})$ for $i = 0, \ldots, m-k-1$.

And we have for $0 \leq i + j < m - 1$ in the commutative ring $Z(A)[\gamma]
\sum_{i=0}^{m-1} \gamma^{(i+1)\ell} \cdot \gamma^{j\ell} = \sum_{i=0}^{m-1} \left(\gamma^{i+j+1}\right)^\ell = \frac{1 - \left(\gamma^{i+j+1}\right)^m}{1 - \gamma^{i+j+1}} = 0.$

Therefore, by Proposition 1 and Remark 2, $\mathcal{L}^\perp \subseteq \mathcal{R}$ and $^\perp \mathcal{R} \subseteq \mathcal{L}$.

Again by Proposition 1 and Remark 2 an element $x \in A^m$ lies in $\mathcal{L}^\perp$ if and only if
\begin{equation}
\begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & \gamma & \cdots & \gamma^{m-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \gamma^{k-1} & \cdots & \gamma^{(k-1)(m-1)}
\end{pmatrix}
\begin{pmatrix}
1 \\
\gamma \\
\gamma^2 \\
\gamma^{2(k-1)}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_m
\end{pmatrix} = 0.
\end{equation}

But in the commutative ring $Z(A)[\gamma]$ the matrix
$$
H = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & \gamma & \cdots & \gamma^{2(k-1)} \\
1 & \gamma^2 & \cdots & \gamma^{2(k-1)} \\
1 & \gamma^{k-1} & \cdots & \gamma^{(k-1)(k-1)}
\end{pmatrix} \in M_{4 \times 4}(Z(A)[\gamma])
$$
is invertible. Therefore $H$ is also invertible in $M_{k \times k}(A)$ and thus induces a group automorphism of $A^k$. If we let $x_H = (x_1, \ldots, x_k), x_U = (x_{k+1}, \ldots, x_m)$, we can rewrite equation (1) as
$$
\begin{pmatrix}
H \vert U \\
x_H \\
x_U
\end{pmatrix}
= 0 \quad \text{and} \quad
\begin{pmatrix}
H \vert 0 \\
x_H \\
x_U
\end{pmatrix}
= -\begin{pmatrix}
0 \\
0 \\
x_U
\end{pmatrix}.
$$

For each choice of $x_U$ we have only one possible value for $x_H$. Thus $|\mathcal{L}^\perp| = |A|^{m-k} = |\mathcal{R}|$ by Proposition 1 and therefore $\mathcal{L}^\perp = \mathcal{R}$. Similarly, we have $^\perp \mathcal{R} = \mathcal{L}$.

$\square$
Theorem 3 Let $\Gamma \in M_{\ell\times\ell}(\mathbb{F}_{q^s})$ be a primitive $m$-th root of unity and $C = Q$-BCH$_q(m,\ell,\delta,\Gamma)$. Then there exists a RRS code $R$ over the ring $M_{\ell\times\ell}(\mathbb{F}_{q^s})$ with parameters $[m, m - \delta + 1, \delta]_{M_{\ell\times\ell}(\mathbb{F}_{q^s})}$ and an $\mathbb{F}_q$-linear, $\mathbb{F}_q$-isometric embedding $\psi : C \to R$.

Proof A parity-check matrix of $C$ is

$$H = \begin{pmatrix} I_{\ell} & \Gamma & \cdots & \Gamma^{m-1} \\ I_{\ell} & \Gamma^2 & \cdots & \Gamma^{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ I_{\ell} & \Gamma^{\delta-1} & \cdots & \Gamma^{(\delta-1)(m-1)} \end{pmatrix} \in M_{(\delta-1)\ell,m\ell}(\mathbb{F}_{q^s}).$$

Remark that $H$ is a generator matrix of the LGRS code with parameters $[x, x, \delta - 1]_{M_{\ell\times\ell}(\mathbb{F}_{q^s})}$ over the ring $M_{\ell\times\ell}(\mathbb{F}_{q^s})$ and by Proposition 3 its dual is the RRS with parameters $[(1, \ldots, 1), x, \delta - 1]_{M_{\ell\times\ell}(\mathbb{F}_{q^s})}$.

Now let

$$\psi : C \to \left( M_{\ell\times\ell}(\mathbb{F}_{q^s}) \right)^m$$

$$(c_{11}, \ldots, c_{1\ell}, \ldots, c_{m1}, \ldots, c_{m\ell}) \mapsto \begin{bmatrix} (c_{11} \ 0 \ \cdots \ 0) \\ \vdots \\ (c_{\ell1} \ 0 \ \cdots \ 0) \end{bmatrix} \begin{bmatrix} (c_{m1} \ 0 \ \cdots \ 0) \\ \vdots \\ (c_{m\ell} \ 0 \ \cdots \ 0) \end{bmatrix}.$$ 

Obviously, $\psi$ is $\mathbb{F}_q$-linear, injective and isometric and by the above remark we have $\psi(C) \subseteq R$. \(\square\)

Theorem 3 generalizes the well-known [15, Theorem 2, page 300] relation between BCH codes and Reed-Solomon codes. The above relation will allow us to adapt the unique decoding algorithm from [1] to quasi-BCH codes.

3.2 The Welch-Berlekamp algorithm for quasi-BCH codes

In this Subsection we let $A$ designate a finite ring with identity. Before giving the Welch-Berlekamp decoding algorithm [4,9], we need to define what the evaluation of a bivariate polynomial over $A$ is. Let $Q = \sum Q_{i,j}X^iY^j \in A[X,Y]$ be such a polynomial. We define the evaluation of $Q$ at $(a,b) \in A^2$ to be

$$(a,b)Q = \sum a^ib^jQ_{i,j} \in A.$$ 

Be careful of the order of $a$, $b$ and $Q_{i,j}$. This choice will be explained in the proof of Lemma 1. Let $f \in A[X]$, we define the evaluation of $Q$ at $f$ to be

$$(X,f(X))Q = \sum X^j(f(X))^jQ_{i,j} \in A[X].$$

As in the univariate case, the evaluation maps defined above are not ring homomorphisms in general.

Lemma 1 Let $g \in A[X]$, $Q \in A[X,Y]$ of degree at most 1 in $Y$ and $a \in A$. Then

$$(a)((X,g(X))Q) = (a,(a)g)Q.$$
Proof We write \(Q(X, Y) = Q_0(X) + Q_1(X)Y = Q_0(X) + (\sum_i Q_{1i}X^i)Y\). The proof is an easy calculation:

\[
(a)((X, g(X))Q) = (a) \left( Q_0(X) + \sum_i X^i g(X)Q_{1i} \right)
\]

\[
= (a)Q_0 + \sum_i a^i (a)gQ_{1i}
\]

\[
= (a, (a)g)Q \text{ by definition.}
\]

\[\square\]

We let \(C = \text{Q-BCH}_q(m, \ell, \delta, \Gamma)\), \(\tau = \left\lfloor \frac{\delta - 1}{2} \right\rfloor\), \(k = m - \delta + 1\) and

\[
\begin{bmatrix}
a_{11}^1 & \cdots & a_{1\ell}^1 \\
a_m^1 & \cdots & a_m^1 \\
\vdots & \ddots & \vdots \\
a_{11}^\ell & \cdots & a_{1\ell}^\ell
\end{bmatrix}
\begin{bmatrix}
\text{pr} : (M_{\ell \times \ell}(\mathbb{F}_q^\tau))^m \rightarrow \mathbb{F}_q^{m\ell}
\end{bmatrix}
\]

\[
\begin{bmatrix}
a_{11} \ldots a_{1\ell} \\
a_m \ldots a_m \\
\vdots \ldots \vdots \\
a_{11} \ldots a_{1\ell}
\end{bmatrix}
\rightarrow
\begin{bmatrix}
a_{11}^1, \ldots, a_{1\ell}^1, a_m^1, \ldots, a_m^\ell
\end{bmatrix}.
\]

Algorithm 1 Welch-Berlekamp for quasi-BCH codes

Input: a received vector \(y \in \mathbb{F}_q^{m\ell}\) with at most \(\tau\) errors.

Output: the unique codeword within distance \(\tau\) of \(y\).

1: \((Z_1, \ldots, Z_m) \leftarrow \psi(y)\) where \(\psi\) is the map from Theorem 3.
2: Find \(Q = Q_0(X) + Q_1(X)Y \in (M_{\ell \times \ell}(\mathbb{F}_q^\tau))[X][Y]\) of degree 1 such that
   1. \((F^{i-1}, Z_i)Q = 0\) for all \(i = 1, \ldots, m\),
   2. \(\deg Q_0 \leq m - \tau - 1\),
   3. \(\deg Q_1 \leq m - \tau - 1 - (k - 1)\).
3: \(f \leftarrow\) the unique root of \(Q\) in \((M_{\ell \times \ell}(\mathbb{F}_q^\tau))[X]_{<\delta}\) such that
   \(d((Z_1, \ldots, Z_m), ((I_1)f, \ldots, (I_m)f)) \leq \tau\).
4: \(\text{return } \text{pr} ((I_1)f, (I)f, \ldots, (I_m^{m-1})f)\).

Lemma 2 Let \(y \in \mathbb{F}_q^{m\ell}\) be a received word containing at most \(\tau\) errors. Then there exists a nonzero bivariate polynomial \(Q = Q_0 + Q_1 Y \in (M_{\ell \times \ell}(\mathbb{F}_q^\tau))[X][Y]\) satisfying

1. \((F^{i-1}, Z_i)Q = 0\) for \(i = 1, \ldots, m\).
2. \(\deg Q_0 \leq m - \tau - 1\).
3. \(\deg Q_1 \leq m - \tau - 1 - (k - 1)\).

Proof We solve the problem with linear algebra over \(\mathbb{F}_q^\tau\). We have, for each column of the solution, \(m\ell\) equations and \(\ell \left\lfloor (m - \tau) + (m - \tau - (k - 1)) \right\rfloor > \ell m\) unknowns by Proposition 1. \(\square\)

Lemma 3 Let \(Q \in (M_{\ell \times \ell}(\mathbb{F}_q^\tau))[X][Y]\) satisfying the three conditions of Lemma 2 and \(f \in (M_{\ell \times \ell}(\mathbb{F}_q^\tau))[X]_{<\delta}\) be such that \(d((Z_1, \ldots, Z_m), ((I_1)f, \ldots, (I_m)f)) \leq \tau\). Then \((X, f(X))Q = 0\).

Proof The polynomial \((X, f(X))Q\) has degree at most \(m - \tau - 1\). By Lemma 1 we have \((F^{i-1})(X, f(X))Q) = (F^{i-1}, (I^{m-1})f)Q) = (F^{i-1}, Z_i)Q = 0\) for at least \(m - \tau\) values of \(i \in \{1, \ldots, m\}\). And therefore we must have \((X, f(X))Q = 0\). \(\square\)
Proposition 4  Algorithm 1 works correctly as expected and can correct up to \(\left\lfloor \frac{\delta - 1}{2} \right\rfloor\) errors.

**Proof** This is a direct consequence of Lemmas 2 and 3. \(\square\)

4 Quasi-BCH codes as interleaved codes

In this Section we prove that quasi-BCH codes can be viewed as an interleaving of classical Reed-Solomon codes. We fix for this Section \(\Gamma \in M_{\ell \times \ell}(\mathbb{F}_q^s)\) a primitive \(m\)-th root of unity and \(\mathcal{C} = \text{Q-BCH}_q(m, \ell, \delta, \Gamma)\).

**Definition 9** Let \(\mathcal{C}_1, \ldots, \mathcal{C}_\ell\) be error correcting codes of length \(m\) over \(\mathbb{F}_q\). The interleaved code \(\mathcal{C}\) with respect to \(\mathcal{C}_1, \ldots, \mathcal{C}_\ell\) is a subset of \(M_{\ell \times m}(\mathbb{F}_q)\), equipped with the \(\ell\)-block distance with respect to the columns, such that \(\mathbf{c} \in \mathcal{C}\) if and only if the \(i\)-th row of \(\mathbf{c}\) is a codeword of \(\mathcal{C}_i\) for \(i = 1, \ldots, \ell\).

**Lemma 4** The matrix \(\Gamma\) diagonalizes over an extension of \(\mathbb{F}_{q^s}\) and its eigenvalues are all primitive \(m\)-th roots of unity.

**Proof** Let \(\mathbb{F}_{q^s} \supseteq \mathbb{F}_{q^s}\) be the splitting field of \(X^m - 1\). The polynomial \(X^m - 1\) is a multiple of the minimal polynomial \(\mu(X)\) of \(\Gamma\). Hence the eigenvalues of \(\Gamma\) are \(m\)-roots of unity. Let \(P \in \text{GL}_\ell(\mathbb{F}_{q^s})\) be such that \(P^{-1} \Gamma P\) is diagonal. Now if an eigenvalue \(\lambda_i\) of \(\Gamma\) has order \(d < m\), then

\[
P^{-1}(\Gamma^d - I_\ell)P = \begin{pmatrix}
\lambda_1^d \\
\vdots \\
\lambda_i^d \\
\vdots \\
\lambda_\ell^d
\end{pmatrix} - I_\ell
\]

is singular as its \(i\)-th diagonal element would be zero. Consequently \(\Gamma^d - I_\ell \notin \text{GL}_\ell(\mathbb{F}_{q^s})\) which is absurd. \(\square\)

**Theorem 4** The quasi-BCH code \(\mathcal{C}\) over \(\mathbb{F}_q\) is an interleaved code of \(\ell\) subcodes of Reed-Solomon codes over \(\mathbb{F}_{q^s}\) in the following sense: there exist \(\ell\) Reed-Solomon codes \(\mathcal{C}_1, \ldots, \mathcal{C}_\ell\) over \(\mathbb{F}_q\) and an isometric isomorphism from \(\mathcal{C}\), equipped with the \(\ell\)-block distance, to a subcode of the interleaved code with respect to \(\mathcal{C}_1, \ldots, \mathcal{C}_\ell\).

**Proof** We take the notation of the proof of Lemma 4. Recall that

\[
H = \begin{pmatrix}
I_\ell & \Gamma & \cdots & \Gamma^{m-1} \\
I_\ell & \Gamma^2 & \cdots & \Gamma^{2(m-1)} \\
\vdots & \vdots & & \vdots \\
I_\ell & \Gamma^{\ell-1} & \cdots & \Gamma^{(\ell-1)(m-1)}
\end{pmatrix} \in M_{(\delta-1)\ell, m\ell}(\mathbb{F}_{q^s})
\]
is a parity check matrix for $\mathcal{C}$ (proof of Theorem 3). Let

$$
\begin{pmatrix}
v_{11} \\
\vdots \\
v_{1\ell} \\
v_{m1} \\
\vdots \\
v_{m\ell}
\end{pmatrix} = \begin{pmatrix} P^{-1} \\
\ddots \\
& P^{-1}
\end{pmatrix}
\begin{pmatrix}
c_{11} \\
\vdots \\
c_{1\ell} \\
c_{m1} \\
\vdots \\
c_{m\ell}
\end{pmatrix}
$$

(2)

and denote by $\sigma$ the application defined by (2). By Lemma 4 we have that $(c_{11}, \ldots, c_{1\ell}, \ldots, c_{m1}, \ldots, c_{m\ell}) \in \mathcal{C} \iff$

$$
\begin{pmatrix} P^{-1} \\
\ddots \\
& P^{-1}
\end{pmatrix}
\begin{pmatrix} I_{\ell} & \Gamma & \cdots & \Gamma^{m-1} \\
I_{\ell} & I^{2} & \cdots & I^{2(m-1)} \\
\vdots & \vdots \\
I_{\ell} & I^{\delta-1} & \cdots & I^{(\delta-1)(m-1)}
\end{pmatrix}
\begin{pmatrix} P \\
\ddots \\
& P
\end{pmatrix}
\times \sigma
\begin{pmatrix}
c_{11} \\
\vdots \\
c_{1\ell} \\
c_{m1} \\
\vdots \\
c_{m\ell}
\end{pmatrix}
= 0
$$

and $(c_{11}, \ldots, c_{1\ell}, \ldots, c_{m1}, \ldots, c_{m\ell}) \in \mathbb{F}_q^{m\ell}$

Therefore we have

$$(c_{11}, \ldots, c_{1\ell}, \ldots, c_{m1}, \ldots, c_{m\ell}) \in \mathcal{C} \iff \sigma^{-1}(v_{11}, \ldots, v_{1\ell}, \ldots, v_{m1}, \ldots, v_{m\ell}) \in \mathbb{F}_q^{m\ell} \text{ and for } i = 1, \ldots, \ell
$$

$$
\begin{pmatrix}
1 & \lambda_i & \cdots & \lambda_i^{m-1} \\
1 & \lambda_i^2 & \cdots & \lambda_i^{2(m-1)} \\
\vdots & \vdots \\
1 & \lambda_i^{\delta-1} & \cdots & \lambda_i^{(\delta-1)(m-1)}
\end{pmatrix}
\begin{pmatrix} v_{1i} \\
\vdots \\
v_{mi}
\end{pmatrix} = 0.
$$

(3)

Then it is straightforward that $\sigma$ is an isometric isomorphism from $\mathcal{C}$ equipped with the $\ell$-block distance and $\sigma(\mathcal{C})$, which is by equation (3) a subcode of the interleaved code with respect to $\ell$ subcodes of Reed-Solomon codes over $\mathbb{F}_q$. For $i = 1, \ldots, \ell$ take $\mathcal{C}_i$ to be the Reed-Solomon code defined by the parity check matrix of equation (3).

\[\square\]

Note that if the minimal polynomial of $\Gamma$ has degree one: $\Gamma = X - \lambda$, then $s' = s$ and $\Gamma$ diagonalizes as $\lambda I_{\ell}$. Consequently the Reed-Solomon codes $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$ are equal, as they are defined by the same control equations in equation (3). In such a case, we can apply the result on the correction capacity for interleaved Reed-Solomon codes [5,18].

**Corollary 1** There exists a decoding algorithm that is guaranteed to correct up to $\frac{\ell-1}{2}$ errors. In particular, if the minimal polynomial of $\Gamma$ has degree 1 over $\mathbb{F}_q$, then it can correct up to $\frac{\ell}{\ell+1}(\delta - 1)$ errors with high probability.
Proof Taking the notation of Theorem 4 and if \( y = c + e \) is a received word, one can decode \( \sigma(y) \) with the decoding algorithms of \( C_1, \ldots, C_\ell \) obtaining \( c' \in \mathbb{F}_q^{\ell} \). Then \( c = \sigma^{-1}(c') \). If the minimal polynomial of \( \Gamma \) has degree 1, then \( C_1 = C_2 = \cdots = C_\ell \) and one can apply the algorithm of [5] or [18]. \( \square \)

5 Acknowledgements

This work has been partly supported by the University of Caen, the University of Joseph Fourier and the french Direction Générale de l’Armement.

References

Quasi-perfect linear codes from plane cubics

Irene Platoni

Received: date / Accepted: date

Abstract We present some recently obtained constructions of linear quasi-perfect codes with small density arising from plane cubic curves.

Keywords Galois affine spaces · Bicovering arcs · Complete caps · Quasi-perfect codes · Cubic curves

Mathematics Subject Classification (2000) 94B27 · 51E20

1 Introduction

Let $\mathbb{F}_q$ be the finite field with $q$ elements. A $q$-ary linear code $C$ of length $n$ and dimension $k$ is a $k$-dimensional linear subspace of $\mathbb{F}_q^n$. The number of non-zero positions in a vector $v \in \mathbb{F}_q^n$ is called the Hamming weight $\omega(v)$ of $v$. For $v_1, v_2 \in \mathbb{F}_q^n$ the Hamming distance $d(v_1, v_2)$ is the weight $\omega(v_1 - v_2)$. The minimum distance of $C$ is

$$d(C) := \min \{ \omega(x) \mid x \in C, x \neq 0 \}$$

and a $q$-ary linear code of length $n$, dimension $k$ and minimum distance $d$ is called an $[n,k,d]_q$-code. Such a code is said to be $t$-error correcting, where $t$ is the integer part of $(d - 1)/2$. The covering radius of $C$ is the minimum integer $R(C)$ such that for any vector $v \in \mathbb{F}_q^n$ there exists $x \in C$ with $d(v, x) \leq R(C)$. An $[n,k,d]_q$-code with covering radius $R$ is denoted by $[n,k,d]_q R$. Clearly, $R(C) \geq t$ holds and when equality is attained the code $C$ is said to be perfect. As there are only finitely many classes of linear perfect codes, of particular interest are those codes $C$ with $R(C) = t + 1$, called quasi-perfect codes (see [5,7,8]). One of the parameters characterizing the covering

Department of Mathematics, University of Trento
Via Sommarive, 14 - 38123 Trento (Italy)
Tel.: +39-0461-283982
Tel.: +39-075-5855039
E-mail: irene.platoni@unitn.it
quality of an \([n,k,d]_q\)-R-code \(C\) is its covering density \(\mu(C)\), introduced in [9] as the average number of codewords at distance less than or equal to \(R\) from a vector in \(\mathbb{F}_q^n\):

\[
\mu(C) = \frac{V_q(n,R)}{q^{n-k}},
\]

where

\[
V_q(n,j) = \sum_{i=0}^j \binom{n}{i} (q-1)^i
\]

is the size of a sphere of radius \(j\) in \(\mathbb{F}_q^n\). The covering density \(\mu(C)\) is always greater than or equal to 1, and equality holds precisely when \(C\) is perfect. Among codes with the same codimension \(s\) and covering radius \(R\), the shortest ones have the best covering density. This motivates the following notation:

\(l(s,R,q)\) is the minimal length \(n\) for which there exists an \([n,n-s,d]_q\)-R-code with given \(s,q,d\) and \(R\).

The problem of estimating \(l(s,R,q)\) has been broadly investigated (see the seminal paper [6], as well as the recent survey [10] and the references therein).

In this talk, we will restrict our attention to codes with covering radius \(R = 2\) and \(d = 4\), i.e. quasi-perfect linear codes that are both 1-error correcting and 2-error detecting.

Interestingly, such codes have a nice geometrical counterpart. In fact, the columns of a parity check matrix of an \([n,n-s,4]_q\)-2-code can be considered as a point set \(C\) in the finite projective space \(PG(s-1,q)\) with the following property: no three points of \(C\) are collinear, and every point in \(PG(s-1,q) \setminus S\) is collinear with two points of \(C\). Such a set is called a complete cap in \(PG(s-1,q)\). The following result holds.

**Proposition 1**

\(l(2,2,q) = \min\{n \mid \text{there exists a complete cap of size } n \text{ in } PG(s-1,q)\}\).

This makes it possible to use methods from both Galois Geometries and Algebraic Geometry in order to investigate covering-radius-2 codes with small density.

Here, we are going to discuss some recently obtained upper bounds on \(l(s,2,q)\) which are valid for arbitrarily large values of \(q\). The key tool is the construction of small complete caps in higher dimensional spaces from plane cubic curves defined over \(\mathbb{F}_q\). We remark that our results are of interest from an asymptotical point of view mainly; this is the reason why specific examples of linear codes are not presented in this abstract.

### 2 Complete caps from bicovering and almost bicovering arcs

An \(n\)-cap in an (affine or projective) Galois space over \(\mathbb{F}_q\) is a set of \(n\) points no three of which are collinear. An \(n\)-cap is said to be complete if it is not contained in an \((n+1)\)-cap. A plane \(n\)-cap is also called an \(n\)-arc. As recalled in Introduction, the problem of constructing quasi-perfect codes with covering radius 2 and small
density translates into the language of Galois Geometries as that of constructing small complete caps in a given projective space $PG(N, q)$.

For the size $t$ (AG $(N, q)$) of the smallest complete cap in the Galois affine space $AG(N, q)$ of dimension $N$ over $\mathbb{F}_q$, the trivial lower bound is $\sqrt{2q^{N+1}}$. General constructions of complete caps whose size is close to this lower bound are only known for $q$ even and $N$ odd, see [11, 14, 18, 22]. When $N$ is even, complete caps of size of the same order of magnitude as $cq^{N/2}$, with $c$ a constant independent of $q$, are known to exist for both the odd and the even order case, see [11, 12, 17, 18, 19].

Let $q$ be an odd prime power and $N \equiv 0 \pmod{4}$. Under these assumptions, small complete caps can be obtained via the product method for caps from bicovery and almost bicovery arcs.

Let $q' = q^{\frac{N-2}{2}}$ and let $c$ be a non-square in $\mathbb{F}_q$. Fix a basis of $\mathbb{F}_{q'}$ as a linear space over $\mathbb{F}_q$, and identify points in $AG(N, q)$ with vectors of $\mathbb{F}_{q'} \times \mathbb{F}_{q'} \times \mathbb{F}_q \times \mathbb{F}_q$. For an arc $A$ in $AG(2, q)$, let

$$K_A = \{ (\alpha, \alpha^2, u, v) \in AG(N, q) \mid \alpha \in \mathbb{F}_{q'}, (u, v) \in A \}.$$ 

The set $K_A$ is a cap in $AG(N, q)$ by [4, Theorem 4]. The completeness of the cap $K_A$ depends on the bicovery properties of $A$ in $AG(2, q)$. To introduce the notion of a bicovery arc, the concepts of a regular and a pseudo-regular point with respect to a complete arc in $AG(2, q)$ are necessary.

According to Segre [24], given three pairwise distinct points $P, P_1, P_2$ on a line $\ell$ in $AG(2, q)$, $P$ is external or internal to the segment $P_1P_2$ depending on whether $(x - x_1)(x - x_2)$ is a non-zero square or a non-square in $\mathbb{F}_q$, where $x, x_1$ and $x_2$ are the coordinates of $P, P_1$ and $P_2$ with respect to any affine frame of $\ell$. This definition can be extended as follows.

**Definition 1** Let $A$ be a complete arc in $AG(2, q)$. A point $P \in AG(2, q) \setminus A$ is said to be regular with respect to $A$ if $P$ is external to any segment cut out by the secants of the arc, passing through $P$. The point $P$ is said to be pseudo-regular with respect to $A$ if it is internal to any segment cut out by the secants of the arc, passing through $P$. Finally the point $P$ is said to be bicovery with respect to $A$ if it is neither regular nor pseudo-regular, i.e. if there exist $P_1, P_2, P_3, P_4 \in A$ such that $P$ is both external to the segment $P_1P_2$ and internal to the segment $P_3P_4$.

**Definition 2** Let $A$ be a complete arc in $AG(2, q)$. If every $P \in AG(2, q) \setminus A$ is bicovery by $A$, then $A$ is said to be a bicovery arc. If there exists precisely one point $Q \in AG(2, q) \setminus A$ which is not bicovery by $A$, then $A$ is said to be almost bicovery, and $Q$ is called the center of $A$.

A key tool to obtain complete caps from bicovery and almost bicovery arcs is the following result from [18].

**Theorem 1** (Propositions 4.2-3-4 in [18])

1) Let $A$ be a bicovery $n$-arc in $AG(2, q)$; then $K_A$ is a complete cap in $AG(N, q)$ of size $nq^{\frac{N-2}{2}}$. 
2) Let $A$ be an almost bicovering arc in $AG(2,q)$ with center $Q = (x_0, y_0)$; then:

i) if $Q$ is regular, the set
$$K = K_A \cup \{ (\alpha, \alpha^2 - c, x_0, y_0) \in AG(N,q) \mid \alpha \in \mathbb{F}_q \}$$
is a complete cap in $AG(N,q)$ of size $(n+1)q^{N-2}$.

ii) if $Q$ is pseudo-regular, the set
$$K = K_A \cup \{ (\alpha, \alpha^2 - c^2, x_0, y_0) \in AG(N,q) \mid \alpha \in \mathbb{F}_q \}$$
is a complete cap in $AG(N,q)$ of size $(n+1)q^{N-2}$.

Results on complete caps in projective spaces can be deduced from results on complete caps in affine spaces, and conversely. Since the affine space $AG(N,q)$ is embedded in the projective space $PG(N,q)$, a complete cap in $AG(N,q)$ can be viewed as a cap in $PG(N,q)$, whose completeness can be achieved by adding some extra-points at the hyperplane at infinity. Therefore, taking into account equality (1), the following relation holds:
$$l(N+1,2,q)_4 \leq t(AG(N,q)) + L(N,2,q)_4.$$

Here $L(N,2,q)_4$ denotes the maximal length $n$ of a quasi-perfect $[n, n-N, 4]_2$-code with codimension $N$ and order $q$ (or, equivalently, the maximal size of a complete cap in $PG(N-1,q)$).

3 Small complete caps from cubic curves

Although complete arcs contained in cubic curves have been deeply investigated by Szőnyi [25,26,27] and Voloch [28] and by Hirschfeld-Voloch [20] since the eighties, these geometrical objects are the base for some recent inductive constructions of complete caps in higher dimensions.

From now on we assume that the characteristic of $\mathbb{F}_q$ is $p > 3$. Let $\mathcal{X}$ be an irreducible plane cubic curve defined over $\mathbb{F}_q$, and consider the set $G$ of the non-singular $\mathbb{F}_q$-rational points of $\mathcal{X}$. As it is well known, for any point $O$ of $G$ it is possible to give a group structure to $G$, by defining a binary operation $\boxplus$ in such a way that $(G, \boxplus)$ is an abelian group with neutral element $O \in G$. The point $O$ is usually chosen as an inflection point. One of the main properties of this operation is that three distinct points in $G$ are collinear if and only if their sum is the neutral element in $G$. So, arcs contained in $G$ can be obtained as follows.

**Theorem 2** Let $H$ be a subgroup of $G$ of index $m$, with $(3,m) = 1$, and let $Q$ be a point in $G \setminus H$. Then the coset $K = H \boxplus Q$ is an arc.

In order to investigate the covering properties of the arc $K$ of Theorem 2, we describe a general method, due to Segre [23] and Lombardo-Radice [21], that uses Hasse-Weil’s Theorem to prove the completeness of arcs contained in conic or cubic curves. This method is based on the following idea for proving that the secants of $K$ cover a generic point $P$ off $K$: 
1. Write $K$ in an algebraically parametrized form; i.e. consider polynomial functions over $\mathbb{F}_q$, say $x(T), y(T)$, in such a way that
$$K = \{(a(t), b(t)) \mid t \in A\},$$
where $A$ can be either $\mathbb{F}_q$ or $\mathbb{F}_q^*$. In the case of cubic curves $\mathcal{X}$, this can be easily done when $\mathcal{X}$ is singular;
2. For a point $P = (u, v)$ off $K$, construct an algebraic curve $\mathcal{C}_P$, defined over $\mathbb{F}_q$, describing the collinearity of two points of $K$ and $P$; an affine equation
$$f_P(X, Y) = 0$$
of $\mathcal{C}_P$ is obtained by imposing that the determinant of the matrix
$$\begin{pmatrix}
    u & v & 1 \\
    a(X) & b(X) & 1 \\
    a(Y) & b(Y) & 1
\end{pmatrix}$$
is equal to zero.
3. Show that $\mathcal{C}_P$ is absolutely irreducible or has at least an absolutely irreducible component defined over $\mathbb{F}_q$; also, compute an upper bound $\bar{g}$ for the genus of $\mathcal{C}_P$ (or of its irreducible $\mathbb{F}_q$-rational component).
4. Apply the Hasse-Weil bound. We recall that the Hasse-Weil bound ensures the existence of at least $q + 1 - 2\sqrt{q} \mathbb{F}_q$-rational points of an absolutely irreducible curve of genus $g$ defined over $\mathbb{F}_q$. Then, if $q$ is large enough with respect to $\bar{g}$, the existence of a suitable $\mathbb{F}_q$-rational affine point $(x_0, y_0)$ of $\mathcal{C}_P$ (or of its irreducible $\mathbb{F}_q$-rational component) is guaranteed. This is sufficient to deduce the collinearity between $P$ and the points $(a(x_0), b(x_0))$ and $(a(y_0), b(y_0))$ of $K$.
5. It is possible that for a few points $P$ the curve $\mathcal{C}_P$ is reducible and does not admit an $\mathbb{F}_q$-rational component. In this case, $P$ is not covered by the secants of $K$. In order to obtain the completeness, it is necessary to extend the arc $K$ with some of such points.

As we pointed out in Section 2, bicovery and almost bicovery arcs in affine planes are a powerful tool to construct small complete caps in $AG(N, q)$ with $q$ odd and $N \equiv 0 \pmod{4}$. However, to establish whether a complete arc is bicovery or almost bicovery can be a difficult task. So far, two different types of irreducible plane cubic curves have been investigated in order to prove the bicovery properties of the associated arcs. The non-singular (or elliptic) case was investigated in [1], while the case of a cubic with a cuspidal rational singularity and a rational inflection point is the object of the preprint [2]. Results on bicovery arcs contained in cubics with a node or cubics with an isolated double point have not appeared in literature yet. These cases are currently under investigation [3]; if some new results were obtained before the Conference, then they would be presented in our talk.

**Theorem 3 (Theorem 1 in [1])** Let $q$ be odd, and let $m$ be a prime divisor of $q - 1$, with $7 < m < \frac{1}{8} \sqrt{q}$. Assume that the cyclic group of order $m$ admits a maximal-3-independent subset of size $s$. Then there exists a bicovery $n$-arc in $AG(2, q)$ with
$$s \cdot \left\lceil \frac{q - 2\sqrt{q} + 1}{m} \right\rceil \leq n \leq s \cdot \left( \left\lceil \frac{q - 2\sqrt{q} + 1}{m} \right\rceil + 31 \right),$$
Corollary 1 (Theorem 2 in [1]) Let \( q \) be odd, and let \( m \) be a prime divisor of \( q - 1 \), with \( 7 < m < \frac{1}{8} \sqrt[3]{q} \). Assume that the cyclic group of order \( m \) admits a maximal-3-independent subset of size \( s \). Then for any positive integer \( N \equiv 0 \pmod{4} \), the following inequality holds

\[
l(N + 1, 2, q)_4 \leq s \cdot q^{\frac{m-1}{3}} \cdot \left( \left\lfloor \frac{q - 2\sqrt{q} + 1}{m} \right\rfloor + 31 \right) + L(N, 2, q)_4.
\]

Corollary 2 ([2]) The point \( Q = (0,0) \) is either regular or pseudo-regular with respect to \( S \), according to whether \( q \equiv 1 \pmod{4} \) or \( q \equiv 3 \pmod{4} \).

Theorem 4 ([2]) Let \( q = p^h \) with \( p > 3 \) a prime. Let \( h' \) be an integer with \( 1 < h' < h \) and set \( m = p^{h'} \). Assume that \( q \geq 144m^3 \). Let \( X' \) be the cubic curve with a cuspidal rational singularity and a rational inflection point, and let \( H \) be as in Theorem 2. Then there exists a set \( S \), union of some cosets of \( H \), which is an almost bicovering arc in \( AG(2,q) \) of size \( k \), with

\[
k = \begin{cases} 
\left\lfloor \frac{2\sqrt{m-3}}{m} q \right\rfloor & \text{if } h' \text{ is even}, \\
\left\lfloor \sqrt{m^3 p + 3mp - 3} \right\rfloor q & \text{if } h' \text{ is odd}.
\end{cases}
\]

The point \( Q = (0,0) \) is either regular or pseudo-regular with respect to \( S \), according to whether \( q \equiv 1 \pmod{4} \) or \( q \equiv 3 \pmod{4} \).

Corollary 2 ([2]) Let \( q = p^h \) with \( p > 3 \) a prime, \( h > 8 \). Let \( N \equiv 0 \pmod{4} \), \( N \geq 4 \). Let \( v_h \) be the integer in \( \{1, \ldots, 8\} \) such that \( v_h \equiv h \pmod{8} \), and let \( t_h \) be the integer in \( \{1, \ldots, 4\} \) such that \( t_h \equiv h \pmod{4} \). Assume that \( p^h > 144 \). Then

\[
l(N + 1, 2, q)_4 \leq \left( 2^{h^6 - \left\lfloor (h/4) - 1 \right\rfloor - 1/2} \right) q^{\frac{N-1}{2}} + L(N, 2, q)_4 \leq 2p^2 \cdot q^{\frac{N}{2}} + L(N, 2, q)_4 \leq 2pq^{\frac{N}{2}} + L(N, 2, q)_4.
\]

Corollary 2 ([2]) Let \( q = p^h \) with \( p > 3 \) a prime, \( h > 8 \). Let \( N \equiv 0 \pmod{4} \), \( N \geq 4 \). Let \( v_h \) be the integer in \( \{1, \ldots, 8\} \) such that \( v_h \equiv h \pmod{8} \), and let \( t_h \) be the integer in \( \{1, \ldots, 4\} \) such that \( t_h \equiv h \pmod{4} \). Assume that \( p^h > 144 \). Then

\[
l(N + 1, 2, q)_4 \leq \left( 2^{h^6 - \left\lfloor (h/4) - 1 \right\rfloor - 1/2} \right) q^{\frac{N-1}{2}} + L(N, 2, q)_4 \leq 2p^2 \cdot q^{\frac{N}{2}} + L(N, 2, q)_4 \leq 2pq^{\frac{N}{2}} + L(N, 2, q)_4.
\]

In particular,

\[
l(5, 2, q)_4 < 2pq^{\frac{5}{2}} + q^2 + 1.
\]
References

A New Class of Optimal Variable-Weight Optical Orthogonal Codes

Jin-Ho Chung · Kyeongcheol Yang

Abstract Optical orthogonal codes (OOCs) are widely used as spreading codes in optical fiber networks. They are classified into constant-weight OOCs (CW-OOCs) and variable-weight OOCs (VW-OOCs). While CW-OOCs support a single quality of service (QoS), VW-OOCs may support multiple QoSs according to the weights of their codewords. In this paper, we present a new class of VW-OOCs of length \((q-1)p\) by extending a known CW-OOC of length \(p\), where \(p\) is an odd prime and \(q\) is a prime power such that \(p = (q-1)(q-2)L + 1\) for a positive integer \(L\). They are optimal with respect to the Yang bound. Moreover, they may have arbitrarily large weight whereas the recently reported VW-OOCs have only some small weights.

Keywords Correlation · optical fiber networks · optical orthogonal codes (OOCs) · variable-weight OOC

1 Introduction

Optical fiber networks use unipolar transmission, in which 1 means signal ‘on’ and 0 signal ‘off’ [1]–[3]. In these systems, optical orthogonal codes (OOCs) are employed as signature codes to accommodate multiple users. They are...
classified into constant-weight OOCs (CW-OOCs) and variable-weight OOCs (VW-OOCs). The former supports a single quality of service (QoS), while the latter may support multiple QoSs according to the weights of codewords.

In [4], Chung, Salehi, and Wei presented a guideline for design of OOCs, and gave some constructions of optimal CW-OOCs. Since then, several constructions for optimal CW-OOCs with respect to the Johnson bound [5] have been reported in the literature [6]–[18]. Later, Yang [19] introduced the concept of VW-OOCs and presented a bound on VW-OOCs. Recently, several VW-OOCs with small weights have been constructed [20]–[25]. However, the weight of an OOC is a major factor on the performance of the optical system employing the OOC [19]. Thus, it is an important problem to construct VW-OOCs with large weight.

In this paper, we present a new class of VW-OOCs of length \((q - 1)p\) by extending a CW-OOC of length \(p\), where \(p\) is an odd prime and \(q\) is a prime power such that \(p = (q - 1)(q - 2)L + 1\). These new VW-OOCs are optimal with respect to the Yang bound [19]. Moreover, they have arbitrarily large weight while the VW-OOCs recently reported in [20]–[25] have only some small weights.

The outline of the paper is as follows. In Section II, we give some preliminaries to VW-OOCs. In Section III, we present a new class of optimal VW-OOCs of length \((q - 1)p\). Finally, we give some concluding remarks in Section IV.

2 Preliminaries

Throughout the paper, we denote by \(\langle x \rangle_y\) the least nonnegative residue of \(x\) modulo \(y\) for an integer \(x\) and a positive integer \(y\). We also denote by \(\mathbb{Z}_n\) the ring of integers modulo \(n\) for a positive integer \(n\). The function \(I[x]\) is defined as \(I[x] = 1\) if \(x\) is true, and 0 otherwise.

Let \(\mathcal{V} \triangleq \{V_0, ..., V_{L-1}\}\) be a set of \(\{0, 1\}\)-sequences of length \(N\), where \(V_i = \{V_i(t)\}_{t=0}^{N-1}\) for \(0 \leq i \leq L - 1\). The crosscorrelation \(\Lambda_{i,i'}(\tau)\) between \(V_i\) and \(V_{i'}\) in \(\mathcal{V}\) is defined as

\[
\Lambda_{i,i'}(\tau) = \sum_{t=0}^{N-1} V_i(t) V_{i'}(\langle t + \tau \rangle_N).
\]

If \(i = i'\), it is called the autocorrelation of \(V_i\). The set \(\mathcal{V}\) is called an \((N, W, \Lambda, \lambda, R)\) VW-OOC with \(W = \{w_1, ..., w_m\}\), \(\Lambda = \{\lambda_1, ..., \lambda_m\}\), and \(R = \{r_1, ..., r_m\}\) if it satisfies the following three conditions:

a) there are exactly \(r_j|V|\) codewords with weight \(w_j\) for \(1 \leq j \leq m\), where \(r_1 + \cdots + r_m = 1\);

b) the autocorrelation of \(V_i\) with weight \(w_j\) in \(\mathcal{V}\) satisfies

\[
\Lambda_{i,i}(\tau) \leq \lambda_a(j), \quad \langle \tau \rangle_N \neq 0
\]

for some positive integer \(\lambda_a(j)\); and
A New Class of Optimal VW-OOCs

c) the crosscorrelation between \( V_i \) and \( V_{i'} \) with \( i \neq i' \) in \( V \) is upper bounded by

\[
A_{i,i'}(\tau) \leq \lambda_c
\]

for some positive integer \( \lambda_c \).

In the particular case that \( \lambda_a(1) = \cdots = \lambda_a(m) = \lambda_c = \lambda \), it will be referred to as an \((N,W,\lambda,A,R)\) VW-OOC for short notation. Note that if \( m = 1 \), an \((N,W,\lambda_c,R)\) VW-OOC becomes an \((N,w,\lambda_a,\lambda_c)\) CW-OOC with \( w = w_1 \) and \( \lambda_a = \lambda_a(1) \).

In [19], Yang presented an upper bound on the set size of an \((N,W,\lambda_c,R)\) VW-OOC by generalizing the Johnson bound [5].

**Theorem 1** ([19]) The size of an \((N,W,\lambda_c,R)\) VW-OOC \( V \) satisfying \( \lambda_a(i) \geq \lambda_c \) for any \( 1 \leq i \leq m \) is upper bounded by

\[
|V| \leq \left\lfloor \frac{(N-1)(N-2)\cdots(N-\lambda_c)}{\sum_{i=1}^{m} r_i w_i(w_i-1)\cdots(w_i-\lambda_c)/\lambda_a(i)} \right\rfloor.
\]

If \( V \) satisfies the bound in Theorem 1 with equality, it is called an *optimal VW-OOC*. The bound can be further simplified for an \((N,W,1,R)\) VW-OOC as follows.

**Corollary 1** The size of an \((N,W,1,R)\) VW-OOC \( V \) is upper bounded by

\[
|V| \leq \left\lfloor \frac{N-1}{\sum_{i=1}^{m} r_i w_i(w_i-1)} \right\rfloor.
\]  

(2)

### 3 New Optimal Variable-Weight OOCs

Let \( p \) be an odd prime such that there exist a prime power \( q \) and a positive integer \( L \) satisfying \( p = (q-1)(q-2)L + 1 \). Assume that \( C \triangleq \{ C_0, \ldots, C_{L-1} \} \) is an optimal \((p,q-1,1,1)\) CW-OOC with respect to the Johnson bound [5], where \( C_i \triangleq \{ C_i(t) \}_{t=0}^{p-1} \) for \( 0 \leq i \leq L - 1 \). The assumption on the existence of \( C \) is valid from the result of Yang [8]. For \( 0 \leq i \leq L - 1 \), let \( \text{supp}(C_i) \) be the support of \( C_i \), that is, the set of the nonzero positions of \( C_i \), given by

\[
\text{supp}(C_i) = \{ u_{i,1}, u_{i,2}, \ldots, u_{i,q-1} \}
\]

where \( 0 \leq u_{i,1} < u_{i,2} < \cdots < u_{i,q-1} \leq p - 1 \) and the \( u_{i,j} \)-th component of \( C_i \) is 1 for \( 1 \leq j \leq q - 1 \). Let \( \mathbb{F}_q \) be the finite field of \( q \) elements and \( \alpha \) a primitive element of \( \mathbb{F}_q \). For a nonzero element \( \beta \) of \( \mathbb{F}_q \), we have \( \beta = \alpha^l \) for an integer \( 0 \leq l \leq q - 2 \).
Construction A: Let \( \eta \) be a one-to-one mapping from \( \mathbb{F}_q \) to \( \mathbb{Z}_q \) with \( \eta(0) = 0 \).

For \( 0 \leq i \leq L - 1 \) and \( j \in \mathbb{F}_q \cup \{ \infty \} \), define \( X_{i,j} \triangleq \{ X_{i,j}(t) \}_{t=0}^{(q-1)p-1} \) as

\[
X_{i,j}(t) = \begin{cases} 1, & \text{if } t_1 = u_{i,\eta(\alpha^t + j)} \in \text{supp}(C_i) \text{ and } \eta(\alpha^t + j) \neq 0 \\ 0, & \text{otherwise} \end{cases}
\]

when \( j \in \mathbb{F}_q \), and

\[
X_{i,\infty}(t) = \begin{cases} 1, & \text{if } t_0 = 0 \text{ and } t_1 \in \text{supp}(C_i) \\ 0, & \text{otherwise} \end{cases}
\]

where \( t_0 = (t)_{q-1} \) and \( t_1 = (t)_p \). The VW-OOC \( \mathcal{X} \) is defined as

\[
\mathcal{X} = \{ X_{i,j} | 0 \leq i \leq L - 1, \ j \in \mathbb{F}_q \cup \{ \infty \} \}.
\]

**Theorem 2** The set \( \mathcal{X} \) in Construction A is an optimal \(( (q-1)p, \{q-2, q-1\}, 1, \left\{ \frac{q+1}{q+1}, \frac{2}{q+1} \right\} \) VW-OOC of size \(( q+1)L \).

Proof. Note that \( \eta(\alpha^t + j) \neq 0 \) for all \( 0 \leq t \leq q-2 \) if and only if \( j = 0 \), which implies that the weight of \( X_{i,0} \) is \( q-1 \) and the weight of \( X_{i,j} \) is \( q-2 \) for \( j \in \mathbb{F}_q \setminus \{0\} \). Moreover, \( X_{i,\infty} \) has weight \( q-1 \). Hence, \( W = \{ q-2, q-1 \} \), \( R = \left\{ \frac{q-1}{q+1}, \frac{2}{q+1} \right\} \), and \( |\mathcal{X}| = (q+1)L \).

For \( 0 \leq \tau \leq (q-1)p-1 \), let \( \tau_0 = (\tau)_{q-1} \) and \( \tau_1 = (\tau)_p \). The correlation \( A_{i,j},(i',j')(\tau) \) between \( X_{i,j} \) and \( X_{i',j'} \) can be written as

\[
A_{i,j},(i',j')(\tau) = \sum_{t_0=0}^{q-2} \sum_{t_1=0}^{p-1} I[t_1 = u_{i,\eta(\alpha^t + j)} \in \text{supp}(C_i)] \\
I[(t_1 + \tau_1)_p = u_{i',\eta(\alpha^{t_0} + j')} \in \text{supp}(C_i')]
\]

\[
= \sum_{t_0=0}^{q-2} I[\tau_1 = u_{i',\eta(\alpha^{t_0} + j')} - u_{i,\eta(\alpha^t + j)} \mod p] \\
I[\alpha^{t_0} + j \neq 0] \cdot I[\alpha^{t_0} + j' \neq 0]
\]

In particular, if \( j, j' \in \mathbb{F}_q \), we have

\[
A_{i,j},(i',j')(\tau) = \sum_{t_0=0}^{q-2} \sum_{t_1=0}^{p-1} I[t_1 = u_{i,\eta(\alpha^t + j)} \in \text{supp}(C_i)] \\
I[(t_1 + \tau_1)_p = u_{i',\eta(\alpha^{t_0} + j')} \in \text{supp}(C_i')]
\]

\[
= \sum_{t_0=0}^{q-2} I[\tau_1 = u_{i',\eta(\alpha^{t_0} + j')} - u_{i,\eta(\alpha^t + j)} \mod p] \\
I[\alpha^{t_0} + j' \neq 0] \cdot I[\alpha^{t_0} + j \neq 0]
\]
where the second equality comes from the fact that there exists exactly one value of \( t_1 \) with \( 0 \leq t_1 \leq p - 1 \) satisfying \( t_1 = u_{i,\eta(\alpha^t_0 + j)} \mod p \) for a fixed \( t_0 \). In order to estimate the correlation between \( X_{i,j} \) and \( X_{i',j'} \) in \( X \), we divide the problem into six cases.

Case i) \( i = i', \quad j, j' \in F_q \), and \( \tau_1 = 0 \): Since \( \tau_1 = 0 \), we have

\[
A_{(i,j), (i',j')}(\tau) = \sum_{t_0=0}^{q-2} I \left[ u_{i,\eta(\alpha^{t_0} + \tau_0 + j')} = u_{i,\eta(\alpha^{t_0} + j)} \mod p \right] \\
\cdot I \left[ \alpha^{t_0} + j \neq 0 \right] \cdot I \left[ \alpha^{t_0 + \tau_0} + j' \neq 0 \right] \\
= \sum_{t_0=0}^{q-2} I \left[ \alpha^{t_0 + \tau_0} + j' = \alpha^{t_0} + j \right] \cdot I \left[ \alpha^{t_0} + j \neq 0 \right] \cdot I \left[ \alpha^{t_0 + \tau_0} + j' \neq 0 \right] \\
= \sum_{t_0=0}^{q-2} I \left[ \alpha^{t_0} (1 - \alpha^{\tau_0}) = j' - j \right] \cdot I \left[ \alpha^{t_0} + j \neq 0 \right] \cdot I \left[ \alpha^{t_0 + \tau_0} + j' \neq 0 \right] \\
\leq \sum_{t_0=0}^{q-2} I \left[ \alpha^{t_0} (1 - \alpha^{\tau_0}) = j' - j \right]
\]

where the second equality comes from the fact that \( \eta \) is a one-to-one mapping. Then, by the primitivity of \( \alpha \), it is easily checked that

\[
A_{(i,j), (i',j')}(\tau) \leq \begin{cases} 
q - 1, & \text{if } (i,j) = (i',j') \text{ and } \tau_0 = 0 \\
1, & \text{otherwise.}
\end{cases}
\]

Case ii) \( i = i', \quad j, j' \in F_q \), and \( \tau_1 \neq 0 \): We have

\[
A_{(i,j), (i',j')}(\tau) = \sum_{t_0=0}^{q-2} I \left[ \tau_1 = u_{i,\eta(\alpha^{t_0} + \tau_0 + j')} - u_{i,\eta(\alpha^{t_0} + j)} \mod p \right] \\
\cdot I \left[ \alpha^{t_0} + j \neq 0 \right] \cdot I \left[ \alpha^{t_0 + \tau_0} + j' \neq 0 \right] \\
\leq \sum_{a=1}^{q-1} \sum_{b=1}^{q-1} I \left[ \tau_1 = u_{i,a} - u_{i,b} \mod p \right] \\
= A_{i,i}(\tau_1)
\]

where \( A_{i,i}(\tau_1) \) denotes the autocorrelation of \( C_i \in C \).
Case iii) $i \neq i'$ and $j, j' \in \mathbb{F}_q$: In this case,

$$A_{(i,j),(i',j')}(\tau)$$

$$= \sum_{t_0=0}^{q-2} I \left[ \tau_1 = u_{i',\eta(\alpha^{t_0}+j')} - u_{i,\eta(\alpha^{t_0}+j)} \mod p \right]$$

$$\cdot I \left[ \alpha^{t_0} + j \neq 0 \right] \cdot I \left[ \alpha^{t_0} + j' \neq 0 \right]$$

$$\leq \sum_{a=1}^{q-1} \sum_{b=1}^{q-1} I \left[ \tau_1 = u_{i',a} - u_{i,b} \mod p \right]$$

$$= A_{i,i}(\tau_1) \leq q-1 \sum_{a=1}^{q-1} I \left[ \tau_1 = u_{i',a} \right]$$

where $A_{i,i}(\tau_1)$ denotes the crosscorrelation between $C_i$ and $C_i'$ in $C$.

Case iv) $j \in \mathbb{F}_q$ and $j' = \infty$: Since $X_{i,j'}(t) = 0$ for all $t$ with $t_0 \neq 0$, we have

$$A_{(i,j),(i',j')}(\tau)$$

$$= \sum_{t_1=0}^{p-1} I \left[ t_1 = u_{i,\eta(1+j)} \right] \cdot I \left[ (t_1 + \tau_1)_p \in \text{supp}(C_i') \right] \cdot I \left[ 1 + j \neq 0 \right]$$

$$\leq \sum_{t_1=0}^{p-1} I \left[ t_1 = u_{i,\eta(1+j)} \right]$$

$$\leq 1.$$

Case v) $j = \infty$ and $j \in \mathbb{F}_q$: Similar to the Proof of Case iv).

Case vi) $j = j' = \infty$: If $\tau_0 = 0$, then

$$A_{(i,j),(i',j')}(\tau)$$

$$= \sum_{t_1=0}^{p-1} I \left[ t_1 \in \text{supp}(C_i) \right] \cdot I \left[ (t_1 + \tau_1)_p \in \text{supp}(C_i') \right]$$

$$= A_{i,i'}(\tau_1) \leq \begin{cases} q-1, & \text{if } i = i' \text{ and } \tau_1 = 0 \\ 1, & \text{otherwise.} \end{cases}$$

If $\tau_0 \neq 0$, it is easily checked that

$$A_{(i,j),(i',j')}(\tau) = 0.$$
right-hand side of (2), we get

\[
\left\lfloor \frac{(q-1)p}{q+1} \cdot (q-2)(q-3) + \frac{2}{q+1} \cdot (q-1)(q-2) \right\rfloor
\]

\[
= \left\lfloor \frac{(q+1)(q-1)^2(q-2)L + (q+1)(q-1)-1}{(q-1)^2(q-2)} \right\rfloor
\]

\[
= (q+1)L
\]

where the first equality comes from the fact that \( p = (q-1)(q-2)L + 1 \). Therefore, \( X \) is optimal with respect to the bound in Theorem 1.

Recently, several constructions for optimal \((V, W, 1, R)\) VW-OOCs have been reported in [20]–[25]. In particular, Wu et al. [22] presented optimal VW-OOCs with \( W = \{3, 4\} \) or \( \{4, 5\} \). In [23], Jiang et al. also constructed some optimal VW-OOCs with \( W = \{3, 4, 5\}, \{3, 4, 5, 6\}, \{3, 4, 5, 6, 7\} \) as well as those with \( W = \{3, 4\}, \{3, 5\}, \{3, 6\}, \{3, 7\}, \{4, 5\} \). Moreover, Buratti et al. [24] gave some optimal VW-OOCs with \( W = \{3, 4, 5\}, \{3, 4, 6\}, \{3, 5, 6\}, \) or \( \{3, 4, 5, 6\} \). Although these VW-OOCs may have arbitrarily large lengths, their weights are restricted to some small values. On the other hand, our construction can be applied to any case where an optimal \( (p, q-1, 1, 1) \) CW-OOC with \( p = (q-1)(q-2)L + 1 \) exists. Some numerical examples of parameters of the new VW-OOCs with \( W = \{6, 7\} \) are given in Table 1.

4 Conclusion

In this paper, we presented a construction for VW-OOCs of length \((q-1)p\), where \( p \) is an odd prime and \( q \) is a prime power satisfying \( p = (q-1)(q-2)L + 1 \) for some positive integer \( L \). They are optimal with respect to the Yang bound. Moreover, our construction can be applied to any case where an optimal \( (p, q-1, 1, 1) \) CW-OOC exists. As a result, we obtained VW-OOCs with arbitrarily large weights, unlike the previously known VW-OOCs.

References

<table>
<thead>
<tr>
<th>Yang's CW-OOC in [8] ((N, w, \lambda_a, \lambda_c))</th>
<th>New VW-OOC in Theorem 2 ((N, W, \lambda, R))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(421, 7, 1, 1)</td>
<td>(2947, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(463, 7, 1, 1)</td>
<td>(3241, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(631, 7, 1, 1)</td>
<td>(4417, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(967, 7, 1, 1)</td>
<td>(6769, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1009, 7, 1, 1)</td>
<td>(7063, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1051, 7, 1, 1)</td>
<td>(7357, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1093, 7, 1, 1)</td>
<td>(7651, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1303, 7, 1, 1)</td>
<td>(9121, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1429, 7, 1, 1)</td>
<td>(10003, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1723, 7, 1, 1)</td>
<td>(12061, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(1933, 7, 1, 1)</td>
<td>(13531, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
<tr>
<td>(2017, 7, 1, 1)</td>
<td>(14119, (6, 7), 1, (\left{\frac{7}{11}, \frac{2}{11}\right}))</td>
</tr>
</tbody>
</table>


Optimal Three-Dimensional Optical Orthogonal Codes and Related Combinatorial Designs

Kenneth W. Shum

Abstract Using channel polarization technique in optical code-division multiple access, we can spread optical pulses in the spatial domain, in addition to the time and frequency domains. The pattern of transmitting optical pulses in these three dimensions are specified by the codewords of a three-dimensional optical orthogonal codes (3-D OOC). In this work, combinatorial designs related to optimal 3-D OOC are discussed and some constructions of 3-D OOC are given.

Keywords Optical orthogonal codes, constant weight codes, group divisible design, generalized Bhaskar Rao designs.

1 Introduction

Let $I_n$ be the set $\{0, 1, \ldots, n - 1\}$ and $\mathbb{Z}_n$ be the ring of residues $\mathbb{Z}/(n\mathbb{Z})$. For positive integers $S$, $W$ and $T$, a codeword of a three-dimensional optical orthogonal code (3-D OOC) is defined as a mapping $X$ from $I_S \times I_W \times \mathbb{Z}_T$ to $\{0, 1\}$. We can represent a codeword in two ways. In the first representation, a codeword is identified with a $3 \times S \times W \times T$ matrix $[X(s, w, t)]$, where $X(s, w, t)$ is the image of $(s, w, t) \in I_S \times I_W \times \mathbb{Z}_T$ under the mapping $X$. Alternately, a codeword can be represent by the support of the mapping, defined as the set

\[ \{(s, w, t) \in I_S \times I_W \times \mathbb{Z}_T : X(s, w, t) = 1\}. \]

We define the Hamming correlation function of two codewords $X$ and $Y$ by

\[
H_{X, Y}(\tau) := \sum_{s=0}^{S-1} \sum_{w=0}^{W-1} \sum_{t=0}^{T-1} X(s, w, t)Y(s, w, t \oplus \tau),
\]

where “$\oplus$” denotes modulo-$T$ addition, and $\tau$ is an integer between 0 and $T - 1$. When $X = Y$ and $\tau = 0$, $H_{X, X}(0)$ is the size of the support of $X$, and is called the Hamming weight of $X$. 

Institute of Network Coding, The Chinese University of Hong Kong
E-mail: wkshum@inc.cuhk.edu.hk
This work was partially supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).
An \( (S \times W \times T; \omega, \lambda_0, \lambda_c) \) 3-D OOC, denoted by \( \mathcal{C} \), is a collection of codewords satisfying following conditions [11]:

1) Hamming weight. \( H_{X,X}(0) = \omega \), for all \( X \in \mathcal{C} \),
2) Auto-correlation. \( H_{X,X}(\tau) \leq \lambda_0 \), for all \( X \in \mathcal{C} \) and \( 1 \leq \tau \leq T - 1 \),
3) Cross-correlation. \( H_{X,Y}(\tau) \leq \lambda_c \), for all \( X, Y \in \mathcal{C}, X \neq Y \), and \( 0 \leq \tau \leq T - 1 \).

3-D OOC can be applied in optical code-division multiple access (OCDMA) in which optical pulses are spread in three dimensions, namely spatial, frequency and time. The spatial channels come from polarization of the optical channel. The spreading over frequency is achieved by transmitting optical pulses with different wavelengths. The spreading over time means transmitting optical pulses in different time slots, which are also known as time chips. The parameters \( S, W \) and \( T \) represent the number of available spatial channels, wavelengths and time chips respectively. If codeword \( X \) is assigned to a user, then this user transmits an optical pulse at spatial channel \( s \), wavelength \( w \), and time chip \( t \) if and only if \( X(s, w; t \mod T) = 1 \). When \( S = W = 1 \), the notion of 3-D OOC reduces to the original optical orthogonal code proposed by Chung et al. [3]. Experimental evaluation of OCDMA with spreading in spatial, frequency, and time domains can be found in [7, 14].

Another application of 3-D OOC is digital watermarking for video signal [15]. In this application, we regard a codeword as a 3-D matrix. The cross-sections of the 3-D matrix in the time axis are associated with the frames of picture in a video. A pixel is marked if and only if the corresponding entry in the 3-D matrix is 1.

**Example 1** The followings eight 3-D matrices form a \((2 \times 2 \times 2, 0, 1)\) 3-D OOC:

\[
\begin{align*}
    s = 0 : & \quad \begin{bmatrix}
        1 & 0 \\
        0 & 0 \\
        1 & 0 \\
        0 & 0 \\
        0 & 0 \\
        0 & 0 \\
        0 & 0 \\
        0 & 0 
    \end{bmatrix} \\
    s = 1 : & \quad \begin{bmatrix}
        1 & 0 \\
        0 & 0 \\
        0 & 1 \\
        0 & 0 \\
        1 & 0 \\
        1 & 0 \\
        0 & 1 \\
        0 & 0 
    \end{bmatrix}
\end{align*}
\]

The first (resp. second) row are the first (resp. second) spatial dimension of the 3-D matrices. In each 2-D matrix, the two rows correspond to the two wavelengths, and the two columns correspond to the two time chips. The supports of the codewords are

\[
\begin{align*}
    \{ (0, 0, 0), (1, 0, 0) \}, \{ (0, 0, 0), (1, 1, 0) \}, \{ (0, 0, 0), (1, 0, 1) \}, \\
    \{ (0, 1, 0), (1, 0, 0) \}, \{ (0, 1, 0), (1, 1, 0) \}, \{ (0, 0, 1), (1, 1, 0) \}, \{ (0, 1, 0), (1, 1, 1) \}.
\end{align*}
\]

A wavelength/time plane in a 3-D matrix is called a spatial plane. A 3-D at-most-one-pulse-per-plane code (AMOPPC) is a 3-D OOC if the every spatial plane in every codeword contains at most one optical pulse, i.e.,

\[
\sum_{w=0}^{W-1} \sum_{t=0}^{T-1} X(s, w, t) \leq 1
\]

for all codewords \( X \) and \( s \in \mathbb{Z}_2 \). If there is exactly one “1” in every spatial plane in every codeword of a 3-D OOC, then it is called a single-pulse-per-plane code (SPPC). If there are more than one pulses per spatial plane in any codeword, then the 3-D OOC
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Parameters $S, W, T$</th>
<th>Code size Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[11]</td>
<td>Prime factors of $T$ are all larger than or equal to $SW$</td>
<td>$SW$</td>
</tr>
<tr>
<td>[11]</td>
<td>Prime factors of $W$ and $T$ are all larger than or equal to $S$.</td>
<td>$S$</td>
</tr>
<tr>
<td>[16]</td>
<td>$S = p, W = p^2 - 1, T = p, p$ prime.</td>
<td>$p^2 - 1$</td>
</tr>
<tr>
<td>[16]</td>
<td>$S = W = p^2 - 1, T = p, p$ prime.</td>
<td>$p^2 - 1$</td>
</tr>
<tr>
<td>[12]</td>
<td>$S = W = T = p$, $p$ prime, $1 \leq r \leq p - 2$.</td>
<td>$S$</td>
</tr>
<tr>
<td>[12]</td>
<td>$S = 4, W = q, T \geq 2$, $q$ is a prime power $\geq 4$.</td>
<td>$S$</td>
</tr>
<tr>
<td>[12]</td>
<td>$S = q + 1, W = q, T = p$, $q$ is a prime power $\geq 4$, $p$ is a prime $&gt; q$.</td>
<td>$S$</td>
</tr>
<tr>
<td>[18]</td>
<td>$S = 3, W$ is even when $T$ is even.</td>
<td>$3$</td>
</tr>
<tr>
<td>[18]</td>
<td>$(S - 1)WT \equiv 0 \mod 2$, $S(S - 1)WT \equiv 0 \mod 3$, $S \equiv 0, 1 \mod 4$ when $T \equiv 2 \mod 4$ and $W \equiv 1 \mod 2$.</td>
<td>$3$</td>
</tr>
<tr>
<td>This paper</td>
<td>$S, W, T$ satisfy the conditions in Construction 3</td>
<td>$4$</td>
</tr>
<tr>
<td>This paper</td>
<td>$S - 1 \leq$ all prime power factors $W$, $S \leq$ all prime factors of $T$</td>
<td>$S$</td>
</tr>
<tr>
<td>This paper</td>
<td>$S = 4$, $W$ and $T$ satisfy the conditions in Construction 5 and 6</td>
<td>$4$</td>
</tr>
</tbody>
</table>

Table 1 Summary of constructions for 3-D OOC.

is called a multiple-pulse-per-plane code (MPPC). Likewise, a 3-D at-most-one-pulse-per-time code (AMOPPC) is a 3-D OOC satisfying

$$\sum_{s=0}^{S-1} \sum_{w=0}^{W-1} X(s, w, t) \leq 1$$

for all codewords $X$ and $t \in \mathbb{Z}_T$. If equality holds in the above inequality for all codewords and all $t$, then we have a single-pulse-per-time code (SPTC). Some explicit constructions for 3-D OOC in [11, 12, 16, 18] are summarized in Table 1.

For fixed Hamming weight, Hamming auto- and cross-correlation requirement, we want to construct 3-D OOC with large number of codewords. A Johnson-type bound on the number of codewords for general 3-D OOC, $C$, is given in [16],

$$|C| \leq \left[ \frac{SW}{\omega} \left[ \frac{SWT - 1}{\omega - 1} \left[ \frac{SWT - 2}{\omega - 2} \left[ \cdots \left[ \frac{SWT - \lambda}{\omega - \lambda} \right] \right] \right] \right] \right].$$

Within the class of AMOPPC, the we have a tighter bound on code size [18]

$$|C| \leq \left[ \frac{SW}{\omega} \left[ \frac{(S - 1)WT}{\omega - 1} \left[ \frac{(S - 2)WT}{\omega - 2} \left[ \cdots \left[ \frac{(S - \lambda)WT}{\omega - \lambda} \right] \right] \right] \right] \right].$$

If we remove all the floor operators in the above inequality, we get

$$|C| \leq \left( \frac{\lambda + 1}{\lambda + 1} \right) W^{\lambda + 1} T^\lambda.$$

(2)
A 3-D AMOPPC with code size attaining equality in (2) is said to be **perfect**.

In this work we focus on perfect 3-D SPPC and AMOPPC with \( \lambda = 1 \), and related combinatorial designs. AMOPPC has the property that the Hamming auto-correlation is identically zero for all time shifts \( \tau \). Also, SPPC has the property that the Hamming weight \( w \) is equal to the number of spatial channels \( S \). Hence, we simplify the notation by referring to an \((S \times W \times T, S, 0)\) 3-D SPPC by \((S \times W \times T, \lambda)\)-SPPC, and \((S \times W \times T, \omega, 0)\) 3-D AMOPPC by \((S \times W \times T, \omega, \lambda)\)-AMOPPC.

We study 3-D AMOPPC from a design-theoretical viewpoint, by considering the supports of the codewords as the base blocks of some combinatorial designs. As the spectrum of parameters of perfect 3-D AMOPPC of weight 3 is completely characterized in [18], we will emphasize on 3-D AMOPPC of weight 4 or more. Combinatorial constructions of 2-D OOC and 1-D OOC can be found in [4, 13, 19–22], and the references therein. The parameters of the new constructions in this paper are listed in the last three rows in Table 1.

### 2 Combinatorial Designs

We review some definitions and results from combinatorial design theory, in particular group divisible design and generalized Bhasker Rao design, which are useful in the construction of 3-D OOC.

Let \( v \) be a positive integer and \( K \) be a set of positive integers. A **group divisible design** (GDD) of order \( v \) and block sizes from \( K \) is a triple \((V, G, B)\), where

1. \( V \) is a set of size \( v \), whose elements are called **points**,
2. \( G \) is a partition of \( V \) into disjoint sets, called **groups**,
3. \( B \) is a collection of subsets in \( V \), called **blocks**, such that
   (a) each block in \( B \) has size in \( K \),
   (b) each block intersects every group in \( G \) in at most one point, and
   (c) any pair of points from two distinct groups is contained in exactly \( \lambda \) blocks of \( B \).

We use the notation \( \text{GDD}(K; v) \) for a group divisible design. The **type** of the multiset \( \{|G| : G \in G\} \) is usually written in an “exponential” notation \( i_1^k \cdot i_2^m \cdot i_3^r \cdots \), which means that there are precisely \( i_j \) groups in \( G \) with size \( j \), for \( j = 1, 2, 3, \ldots \). If all groups have the same size \( m \), the GDD is said to be **uniform**, and the corresponding type can be written as \( m^s \) for some \( s \). When all blocks in a GDD have the same size \( k \), we write \( \text{GDD}(k; v) \) instead. A **transversal design** is a uniform \( \text{GDD}(k; v) \) in which every block intersects every group. A transversal design of type \( m^s \) is denoted by \( \text{TD}(k, m) \), where \( k \) denotes the block size and \( m \) denotes the group size. When every group in \( G \) is a singleton, a \( \text{GDD}(K; v) \) is called a **pairwise balanced design** (PBD). We use the notation \( \text{PBD}(K; v) \) for PBD with block sizes in \( K \). If all blocks in a pairwise balanced design have the same size, we have a **balanced incomplete block design** (BIBD). When \( \lambda = 1 \), the subscript \( \lambda \) will be omitted.

**Example 2** Let \( V = \{1, 2, 3, 4, 5\} \) be the point set, and \( G = \{\{1\}, \{2, 3\}, \{4, 5\}\} \) the group set. Let \( B \) be the collection \( B = \{\{1, 2, 4\}, \{1, 3, 5\}, \{2, 5\}, \{3, 4\}\} \).

Then \((V, G, B)\) is a \( \text{GDD}(\{2, 3\}; 5) \) of type \( 1^1 2^2 \).
Example 3 The following is an example of PBD(\{3, 4\}; 10) from [1, Example I.6.7]. The point set \( \mathcal{V} \) is \{0, 1, \ldots, 9\} and the blocks are
\[
\{1, 4, 9\}, \{1, 5, 8\}, \{1, 6, 7\}, \{2, 4, 7\}, \{2, 5, 9\}, \{2, 6, 8\}, \{3, 4, 8\}, \{3, 5, 7\}, \{3, 6, 9\}, \\
\{0, 1, 2, 3\}, \{0, 4, 5, 6\}, \{0, 7, 8, 9\}.
\]

Let \( G \) be a finite abelian group, and let “\( \infty \)” be a special symbol not in \( G \). A generalized Bhaskar Rao design [5] is an \( n \times b \) array with entries in \( G \cup \{ \infty \} \), such that
1) each row has exactly \( r \) entries in \( G \),
2) each column contains exactly \( k \) entries in \( G \), and
3) for each pair of distinct rows \( (x_1, x_2, \ldots, x_b) \) and \( (y_1, y_2, \ldots, y_b) \), the list
\[
x_1 - y_1 : i = 1, 2, \ldots, b, \ x_i \neq \infty \neq y_i,
\]
contains exactly \( \lambda /|G| \) copies of each element in \( G \). We denote a generalized Bhaskar Rao design by \( (n, k, \lambda; G)\)-GBRD. It is straightforward to see that (i) \( \lambda \) is a multiple of \( |G| \), (ii) \( bk = rn \), and (iii) \( r(k - 1) = \lambda(n - 1) \). Hence, the number of columns of an \( (n, k, \lambda; G)\)-GBRD can be calculated by
\[
b = \frac{n(n - 1)}{k(k - 1)}.
\]

When \( n = k \) and \( \lambda = |G| \) for a cyclic group \( G \), the GBRD is also known as cyclic difference matrix over \( G \). We will denote a cyclic difference matrix over a cyclic group \( G \) by \( (n, |G|)\)-CDM.

Example 4 The following 4 \times 12 array,
\[
\begin{array}{cccccccccccc}
\infty & 0 & 0 & 1 & \infty & 1 & 1 & 0 & \infty & 3 & 3 & 0 \\
0 & \infty & 0 & 0 & \infty & 3 & 3 & 0 & \infty & 1 & 1 \\
0 & 1 & \infty & 1 & 2 & 0 & \infty & 2 & 4 & 0 & \infty & 4 \\
0 & 0 & 1 & \infty & 4 & 3 & 0 & \infty & 2 & 1 & 0 & \infty
\end{array}
\]
is a \( (4, 3, 6; \mathbb{Z}_4)\)-GBRD.

We note that if we replace the special symbol \( \infty \) in a GBRD by 0 and the group elements by 1, then the resulting matrix is the incidence matrix of a BIBD. The GBRD is said to be obtained from the corresponding BIBD by “signing” the incidence matrix by the elements of group \( G \). If we start from a group divisible design and sign the associated incidence matrix by the elements of an abelian group, satisfying similar properties as in GBRD, then the resulting matrix is called a generalized Bhaskar Rao group divisible design (GBRGDD) [9]. In the sequel we are interested in signing uniform GDD. The GBRGDD obtained by signing a GDD\(_\lambda(K; ms)\) of type \( m^s \) over a finite abelian group \( G \) is denoted by \( (K, \lambda; G)\)-GBRGDD of type \( m^s \). If \( K = \{k\} \), the GBRGDD has \( ms \) rows and \( \frac{k(k - 1)}{2} m^2 |G| \) columns. We use the notation \( (k, \lambda; G)\)-GBRGDD of type \( m^s \) when \( K \) is the singleton \( \{k\} \).

The notion of GBRGDD naturally encompasses the definitions of GBRD and GDD. When the abelian group \( G \) is the trivial group consisting of a single element, then GBRGDD reduces to a GDD. When the type of GBRGDD is \( 1^s \) for some \( s \), it degenerates to a GBRD.

We record some known results below.
Theorem 1 \[1, 8.5.c\] If the prime power factorization of \(m\) is
\[m = p_1^{e_1} p_2^{e_2} \cdots p_x^{e_x}\]
for some distinct primes \(p_1, p_2, \ldots, p_x\), then there exists a TD\((k, m)\) with
\[k = \min\{e_1, e_2, \ldots, e_x\} + 1.\]

The next theorem characterizes the existence of uniform GDD of block size four.

Theorem 2 \[2\] Let \(m\) and \(s\) be positive integers. A necessary and sufficient condition for the existence of uniform GDD\(_\lambda(4; ms)\) of type \(m^s\) is that the design is not TD\((4, 2)\) and not TD\((4, 6)\), and that
\[
\begin{align*}
\lambda(s-1)m &\equiv 0 \pmod 3, \\
\lambda(s-1)m^2 &\equiv 0 \pmod {12}, \text{ and} \\
\lambda(s-1)m^2 &\equiv 0 \pmod {12}, \text{ and} \\
s &\geq 4.
\end{align*}
\]

For a positive integer \(n\) with prime factorization \(n = p_1 p_2 p_3 \cdots p_c\), an *elementary abelian group* of order \(n\) is defined as the direct product of \(\mathbb{Z}_{p_i}\) for \(i = 1, 2, \ldots, c\), and is denoted by \(EA(n)\). Conditions for the existence of GBRD of block size four signed over elementary abelian group is given in the next theorem.

Theorem 3 \[9, \text{Thm 4.13}\] A necessary condition for a \((4, 4, \lambda; EA(n))\)-GBRD to exist is that \(n\) divides \(\lambda\). If \(n\) divides \(\lambda\), then a \((4, 4, \lambda; EA(n))\)-GBRD exists unless
1. \(n \equiv \lambda \equiv 2 \pmod 4\) when \(n\) is even,
2. \(n = \lambda = 3\) when \(n\) is odd.

The next theorem gives some sufficient conditions for the existence of cyclic difference matrix.

Theorem 4 \[6, 8\] Let \(n \geq 5\) be an odd integer. There exists a \((4, n)\)-CDM when
1. \(\gcd(n, 27) \neq 9\), or
2. \(n = 9i\) for \(5 \leq i \leq 31\).

By adapting Wilson’s fundamental construction of pairwise balanced design \[23\], we have the following product construction of GBRGDD, which is in fact a special case of a more general theorem in \[9\]. Nevertheless, this special case is enough for the subsequent constructions.

Theorem 5 \[9, \text{Thm 5.2}\] Let \(H\) be a subgroup of a finite abelian group \(G\). Suppose that is a \((K_1, \lambda_1; G/H)\)-GBRGDD of type \(m_1^{k_1}\), and a \((K_2, \lambda_2; H)\)-GBRGDD of type \(m_2^{k_2}\) for all \(k_1 \in K_1\). Then there is a \((K_2, \lambda_1 \lambda_2; G)\)-GBRGDD of type \((m_1 m_2)^a\).
Using Theorem 5, we can construct cyclic difference matrix recursively. Recall that a \((k, n, \mathbb{Z}_n)\)-GBRGDD of type 1 is the same as a \((k, s)\)-CDM. When applied to this special case, Theorem 5 says that from two cyclic difference matrices \((k, r)\)-CDM over \(\mathbb{Z}_r\) and \((k, s)\)-CDM over \(\mathbb{Z}_s\), we can construct a \((k, rs)\)-CDM over \(\mathbb{Z}_{rs}\). The idea is essentially the same as the Kronecker-product construction in [17].

**Theorem 6** [17] If the prime factorization of \(T\) is \(p_1 p_2 \cdots p_e\) for (not necessarily distinct) primes \(p_1, p_2, \ldots, p_e\), then there exists a \((k, T)\)-CDM for \(k = \min\{p_1, p_2, \ldots, p_e\}\).

**Proof** For \(i = 1, 2, \ldots, e\), we have cyclic difference matrix \((k, p_i)\)-CDM, constructed from the first \(k\) rows of the mod \(p_i\) multiplication table for example. By combining them using the product construction in Theorem 5, we have a \((k, T)\)-CDM. \(\Box\)

### 3 Combinatorial Constructions of Perfect AMOPPC and SPPC

The main result of this work is the following connection between perfect 3-D AMOPPC and GBRGDD signed over cyclic group.

**Theorem 7** The followings are equivalent:

1. A perfect \((S \times W \times T; !; 1)\)-AMOPPC,
2. \((\omega, T; \mathbb{Z}_T)\)-GBRGDD of type \(W^S\).

By “equivalent” we mean that given a perfect \((S \times W \times T; !; 1)\)-AMOPPC we can construct an \((\omega, T; \mathbb{Z}_T)\)-GBRGDD of type \(W^S\), and vice versa.

We omit the proof of Theorem 7, but give an example instead.

**Example 5** The GBRGDD associated with the 3-D SPPC in Example 1 is the following \((4, 2; \mathbb{Z}_2)\)-GBRGDD of type \(2^2\):

\[
\begin{array}{cccccccc}
0 & 0 & 0 & \infty & \infty & \infty & \infty & \\
\infty & \infty & \infty & 0 & 0 & 0 & 0 & \\
0 & \infty & 1 & \infty & 0 & \infty & 1 & \\
\infty & 0 & \infty & 1 & \infty & 0 & \infty & \\
\end{array}
\]

The first two rows correspond to the first group, while the last two rows correspond to the second. Each column is associated with a codeword of the 3-D OOC.

Utilizing the equivalence between GBRGDD and perfect AMOPPC, we have the following product construction from Theorem 5.

**Construction 1** Given a perfect \((S \times W_1 \times T_1; k, 1)\)-AMOPPC and a perfect \((k \times W_2 \times T_2; \omega, 1)\)-AMOPPC, we can construct a perfect \((S \times W_1 W_2 \times T_1 T_2; \omega, 1)\)-AMOPPC.

Given a GBRD signed over the direct sum of two abelian groups, we can a construct perfect 3-D AMOPCC as well.

**Theorem 8** Let \(G = H \times \mathbb{Z}_T\) be the direct product of an abelian group \(H\) of size \(W\) and a cyclic group \(\mathbb{Z}_T\) of size \(T\). If there exists an \((S, \omega, WT; G)\)-GBRD, then there exists a perfect \((S \times W \times T; \omega, 1)\)-AMOPPC.
Proof (sketch) Let $M$ be an $(S, \omega, WT; H \times \mathbb{Z}_T)$-GBRD. The size of $M$ is $S \times WTS(S - 1)/(\omega(\omega - 1))$. Each column of $M$ has $\omega$ entries in group $G$. Consider a particular column. The rows of the $\omega$ entries specify the spatial channels. Let the content the $\omega$ entries in $G$ be $(w_i, t_i) : i = 1, 2, \ldots, \omega$. By “wavelength shifting”, we construct $W$ codewords with supports $\{(w_i + x, t_i) : i = 1, 2, \ldots, \omega\}$, for $x$ running over the elements in $H$. Repeating the above procedure for each columns of $M$, we obtain $W^2TS(S - 1)/(\omega(\omega - 1))$ codewords, which form a perfect $(S \times W \times T, \omega, 1)$-AMOPPC. 

In Theorem 5, if the first GBRGDD is a GDD($\omega, WS$) of type $WS$ (by taking $K_1 = \{\omega\}$, $G = H$ so that $G/H$ is the trivial group of size 1, $\lambda_1 = 1$, and $m_1 = W$), and the second GBRGDD is an $(\omega, T)$-CDM (by taking $K_2 = \{\omega\}$, $\lambda_2 = T$, $H = \mathbb{Z}_T$, and $m_2 = 1$), we obtain

Construction 2 If there exists a GDD($\omega, WS$) of type $WS$ and an $(\omega, T)$-CDM, then there exists a perfect $(S \times W \times T, \omega, 1)$-AMOPPC.

Combining Theorems 2 and 4 and Construction 2, we get

Construction 3 Let $S$ and $W$ be positive integers satisfying
1. $(S - 1)W \equiv 0 \pmod{3}$,
2. $S(S - 1)W^2 \equiv 0 \pmod{12}$,
3. $S \geq 4$, and
4. $(4, 2) \neq (S, W) \neq (4, 6)$,
and $T \geq 5$ be odd integer satisfying
1. $\gcd(T, 27) \neq 9$, or
2. $T = 9i$ for $5 \leq i \leq 31$,
then there exists a perfect $(S \times W \times T, 4, 1)$-AMOPPC.

We can improve the parameters in the construction of SPPC by Kim et al. in [11].

Construction 4 Let the prime power factorization of $W$ be $W = q_1q_2\cdots q_x$, and the prime factorization of $T$ be $T = p_1p_2\cdots p_y$ ($q_1, q_2, \ldots, q_x$ are distinct prime powers and $p_1, p_2, \ldots, p_y$ are not necessarily distinct primes). If

$$S \leq \min\{q_1 + 1, q_2 + 1, \ldots, q_x + 1, p_1, p_2, \ldots, p_y\},$$

then there exists a perfect $(S \times W \times T, 1)$-SPPC.

Proof Since $S \leq q_i + 1$ for all $i$, we can construct a transversal design $TD(S, W)$ by Theorem 1. Since $S \leq p_j$ for all $j$, we have a cyclic difference matrix $(S, T)$-CDM by Theorem 6. By Construction 2, we obtain a perfect $(S \times W \times T, 1)$-SPPC.

4 Single-pulse-per-plane code with $S = 4$

In this section, we collect some constructions for the special case of SPPC with four spatial channels.
Example 6 From the following $({4, 4, 4}; Z_2 \times Z_2)$-GBRD taken from [10]

\[
\begin{bmatrix}
00 & 00 & 00 & 00 \\
00 & 10 & 10 & 11 \\
00 & 10 & 11 & 01 \\
00 & 11 & 01 & 10 \\
\end{bmatrix}
\]

we can construct a $(4 \times 2 \times 1)$-SPPC by Theorem 8. The supports of the eight codewords are

\[
\{(1, 0, 0), (2, 0, 0), (3, 0, 0), (4, 0, 0)\}, \{(1, 1, 0), (2, 1, 0), (3, 1, 0), (4, 1, 0)\},
\]

\[
\{(1, 0, 0), (2, 0, 1), (3, 1, 0), (4, 1, 1)\}, \{(1, 1, 0), (2, 1, 1), (3, 0, 0), (4, 0, 1)\},
\]

\[
\{(1, 0, 0), (2, 1, 0), (3, 1, 1), (4, 0, 1)\}, \{(1, 1, 0), (2, 0, 0), (3, 0, 1), (4, 1, 1)\},
\]

\[
\{(1, 0, 0), (2, 1, 1), (3, 0, 1), (4, 1, 0)\}, \{(1, 1, 0), (2, 0, 1), (3, 1, 1), (4, 0, 0)\}.
\]

Example 7 From the $(4, 4, 8; Z_2 \times Z_4)$-GBRD from [10]

\[
\begin{bmatrix}
00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\
00 & 01 & 02 & 03 & 10 & 11 & 12 & 13 \\
00 & 03 & 13 & 12 & 11 & 10 & 02 & 01 \\
00 & 12 & 03 & 11 & 13 & 01 & 10 & 02 \\
\end{bmatrix}
\]

we can construct a $(4 \times 2 \times 4, 1)$-SPPC and a $(4 \times 4 \times 2, 1)$-SPPC by Theorem 8.

More generally we have the following construction for even $T$.

Construction 5

1. For $T = 2$, there exists a perfect $(4 \times W \times 2, 1)$-SPPC for all even $W$.
2. For even $T \geq 4$, there exists a perfect $(4 \times W \times T, 1)$-SPPC if $W$ is even and $WT \neq 2^e 3$ for any exponent $e$.

Proof The first part of the construction $T = 2$ follows from Theorem 3 and Construction 8.

For the second part, let $W = 2^{e+1}i$ and $T = 2^{e+2}j$, where $i$ and $j$ are odd integers. By assumption, $ij \neq 3$. From Theorem 3 and Construction 1, we have a $(4 \times i \times j, 1)$-SPPC. By combining several copies of the SPPC in Examples 6 and 7, and applying Construction 1 again, we have a $(4 \times W \times T, 1)$-SPPC. \square

The last construction is for odd $T$.

Construction 6

1. For $T = 3$, there exists a perfect $(4 \times W \times 3, 1)$-SPPC for all $W \neq 2 \mod 4$.
2. For odd $T \geq 5$ satisfying $\gcd(T, 27) = 9$ or $T = 9i$ for $5 \leq i \leq 31$, there exists a perfect $(4 \times W \times T, 1)$-SPPC for $3 \leq W \neq 6$.

Proof Suppose $T = 3$ and $W \neq 2 \mod 4$. We have a $(4, 4, 3W; EA(W) \times Z_3)$-GBRD by Theorem 3. We can then construct a perfect $(4 \times W \times 3, 1)$-SPPC by Theorem 8.

The second part follows directly from Construction 3 by setting $S = 4$. \square
References

7. Garg, M.: Performance analysis and implementation of three dimensional codes in optical code division multiple access system. Master, Thapar University, Punjab, India (2012)
Further Non-randomness in RC4, RC4A and VMPC

Santanu Sarkar

Abstract In this paper we identify several new biases for RC4, RC4A and VMPC, which are designed in similar paradigm. Naturally, these biases provide new distinguishers for the pseudo-random keystream generated from these algorithms. In particular, our result provides the strongest distinguisher against VMPC.

Key-words: Bias, Cryptanalysis, Distinguisher, RC4, RC4A, Stream Ciphers, VMPC.

1 Introduction

Over the last three decades of research and development in stream ciphers, a number of designs have been proposed and analyzed by the cryptology community. One of the main ideas for building a stream cipher relies on constructing a *pseudorandom permutation* and thereafter extracting *pseudorandom words* from this permutation. Till date, the most popular stream cipher among the cryptologists has been RC4 (or alleged RC4). Apart from being used in network protocols such as SSL, TLS, WEP and WPA, the cipher also uses in Microsoft Windows, Apple AOCE, etc.

RC4 is a byte oriented stream cipher where each keystream word is of $n = 8$ bits in size. It has an internal state which is a permutation of all possible $n$-bit integers, that is, a permutation of length $N = 2^n$ (typically 256) words. The size of the secret key $K$ is 5 to 30 bytes in general. This is used to produce the initial permutation (claimed pseudorandom) through a key scheduling algorithm (KSA). Then from the internal state the pseudorandom generation algorithm (PRGA) produces output bytes which get XOR-ed with the plaintext bytes to generate ciphertexts.

RC4 has faced rigorous analysis over the last two decades due to its simple structure. In 2001, Mantin and Shamir [7] observed that the 2nd byte of RC4 keystream is biased towards zero with a probability $2/N$, twice that in case of a random byte-stream. This bias arises from the non-random byte-extraction routine of RC4 PRGA. The bias produces a distinguisher of complexity $O(N)$ for RC4.

In SAC 2010, Sepehrdad et al. [13] presented many empirical biases on the initial keystream bytes, state variables and secret key of RC4. Recently in FSE 2011, Maitra et al. [6] proved that all the initial bytes from 3 to 255 have positive bias towards zero.

Note that RC4 is not suitable for 16/32 bit architecture as output keystream in RC4 is 8-bit. Also there are several distinguishing attacks. To overcome these problems, several variants of RC4 have been proposed. Many of them are much faster than RC4. All these ciphers are based on arrays, modular addition, rotation/swap and memory access. Among them Py is the most popular one. It was designed by Biham et al. [1] and it was selected in the phase 2 of the eStream portfolio. A weakness of this cipher was observed in [11]. In FSE 2004 [10], Paul et al. proposed another variant of RC4 called RC4A. Other popular RC4 variants are Py6 [2], IA [4], NGG [9], GGHN [3], VMPC [16], etc. In Asiacrypt 2006, Paul and Preneel [12] studied weakness of these array based stream ciphers.

RC4A was designed to resist most known attacks on RC4. In [8] the bias of the second output keystream byte of RC4A was exploited. Using this bias, RC4A was distinguished from a random source using $2^{58}$ keystream bytes. The bias between first and third output in RC4A was presented in [15] where the distinguishing attack required $2^{24}$ keystream bytes.

In FSE 2004 [16], another RC4 variant VMPC (Variably Modified Permutation Composition) was proposed. A distinguishing attack on VMPC was first presented in [8] that required $2^{32}$ keystream bytes. Later another distinguishing attack on VMPC was proposed in [15]. This attack requires $2^{40}$ keystream bytes.

1.0.1 Notation

We use the following notation throughout this paper. For round $t \geq 1$ of RC4 PRGA, we denote the indices by $i_t, j_t$, the output keystream byte by $Z_t$, and the permutations before and after the swap by $S_{t-1}$ and $S_t$ respectively. The output bytes at later rounds of RC4 are denoted as $Z_{kN+l}$ for integers $k \geq 0$ and $0 \leq l \leq N - 1$. The initial permutation of RC4 PRGA will be denoted by $S_0$.

1.1 Contribution and organization of the paper

- In FSE 2001 [7], Mantin and Shamir proved that $P(Z_2 = 0) \approx \frac{2}{N}$ in RC4, whereas it should have been $\frac{1}{N}$ for an ideal cipher. So if the probability $P(Z_2 = x)$ is the same for all other non-zero values of $x$, then $P(Z_2 = x) \approx \frac{1-\frac{2}{N}}{\frac{N-2}{N}}$, which is 0.003891 for $N = 256$. In Section 2, We have proved $P(Z_2 = 2) \approx \frac{1}{N} - \frac{3}{N^2}$, which is 0.003860 for $N = 256$.
- After that we have presented a new weakness of RC4A. We have proved that the second byte produced by RC4A has a positive bias towards 2 in Section 3.
- Next in Section 4, we propose a new distinguisher of VMPC. We can distinguish VMPC from a random source using $N^3$ output samples. This is the best distinguishing attack on VMPC. In Table 2 (Section 4), we compare our result on VMPC with the existing works.
Further Non-randomness in RC4, RC4A and VMPC

2 Negative bias in $Z_2$ towards two

RC4 consists of two major components, the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). The internal permutation is of $N$ bytes, and so is the key $K$. However, the original secret key is of length typically between 5 to 30 bytes, and is repeated to form the expanded key $K$. The KSA turns the secret key $K$ into a permutation $S$ of $0, 1, \ldots, N-1$. After that PRGA uses this permutation to generate output keystream. Any addition used in the RC4 description in this paper, is in general addition modulo $N$ unless specified otherwise.

The RC4 algorithm is as shown in Fig. 1.

Now we will show that the probability of $Z_2 = 2$ is $\frac{1}{N} - \frac{3}{N^2}$.

**Theorem 1** Assume that the initial permutation $S_0$ of RC4 PRGA is randomly chosen from the set of all permutations of $\{0, 1, \ldots, N-1\}$. Then the probability that the second output byte of RC4 keystream is 2 is approximately $1/N - 3/N^2$.

**Proof** We will prove that if $S_0[1] = 2$ or $S_0[2] = 0$ or $S_0[2] = 2$, $Z_2$ can not be two.


- $S_0[2] = 0$: Now consider the situation when $S_0[2] = 0$. If $S_0[1] \neq 2$, from the analysis of [7] we know that $Z_2$ will be 0 always. If $S_0[1] = 2$, from the previous analysis $Z_2 \neq 2$.


Now consider the event $E$.

$$E = \{S_0[1] = 2\} \cup \{S_0[2] = 0\} \cup \{S_0[2] = 2\}.$$  

Hence $P(E) = \frac{3}{N} - \frac{1}{N(N-1)} \approx \frac{3}{N}$, as the events $\{S_0[1] = 2, S_0[2] = 2\}$ and $\{S_0[2] = 0, S_0[2] = 2\}$ are impossible. In the case of $E^c$, we assume $P(Z_2 = 2)$ occurs with
\[ \frac{1}{N} \] due to random association. 

Hence,

\[
P[Z_1 = 2] = P[Z_2 = 2 \mid E] \cdot P[E] + P[Z_2 = 2 \mid E^c] \cdot P[E^c] \\
\approx 0 \cdot \frac{3}{N} + \frac{1}{N} \cdot \left(1 - \frac{3}{N}\right) = \frac{1}{N} - \frac{3}{N^2}.
\]

\[
\frac{1}{N} - \frac{3}{N^2}
\]

To the best of our knowledge, except the bias \( Z_2 = 0 \), all the known biases in the keystream are less or equal to \( \frac{1}{N^2} \). Hence the bias presented in this case is the second maximum bias in this direction. Experiment with 1 billion random keys demonstrate that \( P(Z_2 = 2) = 0.003860 \) (away from \( \frac{1}{256} = 0.003906 \)). These conform to the theoretical value. In Figure 2, we have plotted the experimental values of \( P(Z_2 = x) \) for \( x \in [1, 255] \).

\[\text{Fig. 2} \quad \text{Distribution of } P(Z_2 = x) \text{ for } x \in [1, 255].\]

### 3 New bias of RC4A

RC4A is another variant of RC4. In this cipher, two arrays \( S^{(1)}, S^{(2)} \) are used instead of one array. From one chosen key \( k_1 \), another key \( k_2 \) is generated from a pseudorandom bit generator using \( k_1 \) as the seed. Now applying KSA of RC4, two arrays \( S^{(1)}, S^{(2)} \) are generated using the keys \( k_1 \) and \( k_2 \) respectively.

In keystream generation process two variables \( j_1, j_2 \) are used corresponding to \( S^{(1)}, S^{(2)} \). The index pointer \( t_1 = S^{(1)}[i] + S^{(1)}[j_1] \) points to \( S^{(2)} \) instead of \( S^{(1)} \).
Further Non-randomness in RC4, RC4A and VMPC

RC4A PRGA

\[ i = i + 1 \]
\[ j_1 = j_1 + S^{(1)}[i] \]
\[ \text{Swap } S^{(1)}[i] \leftrightarrow S^{(1)}[j_1] \]
\[ Z^{(1)} = S^{(2)}[S^{(1)}[i] + S^{(1)}[j_1]] \]
\[ j_2 = j_2 + S^{(1)}[i] \]
\[ \text{Swap } S^{(2)}[i] \leftrightarrow S^{(2)}[j_2] \]
\[ Z^{(2)} = S^{(1)}[S^{(2)}[i] + S^{(2)}[j_2]] \]

Fig. 3 Pseudo-Random Generation Algorithm of RC4A.

Similarly index pointer \( t_2 = S^{(2)}[i] + S^{(2)}[j_2] \) points to \( S^{(1)} \). The PRGA of RC4A is as shown in Fig. 3.

Now we will prove that second byte generated by RC4A has positive bias towards 2. In fact we will show that \( P(Z^{(2)}_1 = 2) \approx \frac{1}{N} \). To prove our result, we shall need the following result due to [5].

**Theorem 2** Assume initial permutation \( S \) is taken uniformly from the set of all possible permutation of the set \( \{0, 1, \ldots, N - 1\} \). Then for the first index toucher \( t \), we have

\[ P(t = 2) = \frac{2}{N} - \frac{1}{N(N-1)}. \]

Another result of RC4 which will be used in proving our result is from [14].

**Lemma 1** After the first round of RC4 PRGA, the probability \( P(S[1][u] = u) \) is:

\[ P(S[1][u] = u) = \begin{cases} 
0 & \text{if } u = 1, \\
\sum_{X \neq u} P(S[0][1] = X \land S[0][X] = 1) & \text{if } u \neq 1.
\end{cases} \]

Now we will prove the following simple lemma.

**Lemma 2** \( P(Z^{(2)}_1 = t_2) \approx \frac{2}{N} \).

**Proof** Assume \( P(S[0][1][X] = Y) = \frac{1}{N} \) for \( 0 \leq X, Y \leq N - 1 \).

So for \( u \neq 1 \)

\[ P(S^{(1)}_0[u] = u) = P(S^{(1)}_0[1] = u) + \sum_{X \neq u} P(S^{(1)}_0[1] = X \land S^{(1)}_0[u] = u) \]

\[ \approx \frac{1}{N} + \frac{N-1}{N^2} \]

\[ \approx \frac{2}{N}. \]

Using similar approach for \( u = 1 \), it can be proved \( P(S^{(1)}_0[u] = u) \approx \frac{2}{N} \).

Since \( Z^{(2)}_1 = S^{(1)}_1[t_2] \), \( P(Z^{(2)}_1 = t_2) \approx \frac{2}{N} \). \( \Box \)
Now we will prove the following result on RC4A.

**Theorem 3** The probability that the second output byte is equal to two is

\[ P(Z_1^{(2)} = 2) \approx \frac{1}{N - 1}. \]

**Proof** Denote the index toucher of second output byte by \( t_2 \).

\[
P[Z_1^{(2)} = 2] = P[Z_1^{(2)} = 2 \land t_2 = 2] + \sum_{x=0}^{N-1} P[Z_2 = 2 \land t_2 = x].
\]

Now from Theorem 2 and Lemma 2,

\[
P[Z_1^{(2)} = 2 \land t_2 = 2] = P[Z_1^{(2)} = t_2 \mid t_2 = 2] P(t_2 = 2) \approx \frac{2}{N} \cdot \frac{2}{N} = \frac{4}{N^2}
\]

Also for \( x \neq 2 \),

\[
P[Z_1^{(2)} = 2 \land t_2 = x] = P[t_2 = x] P[Z_1^{(2)} = 2 \mid t_2 = x] \approx \frac{1 - \frac{2}{N}}{N - 1} P[S_1^{(1)}[x] = 2] = \left( \frac{1 - \frac{2}{N}}{N - 1} \right)^2 \left( \text{as } P[S_1^{(1)}[2] = 2] \approx \frac{2}{N} \right)
\]

Hence, \( P[Z_1^{(2)} = 2] \approx \frac{4}{N^2} + \left( \frac{1 - \frac{2}{N}}{N - 1} \right)^2 = \frac{1}{N^2}. \) \qed

Note that \( \frac{1}{N^2} > \frac{1}{N} + \frac{1}{N^2} \). Hence with a sample size of \( N^3 \), one can distinguish RC4A from a random source. So, when \( N = 256 \), distinguishing attacks require less than \( 2^{24} \) output samples. In Table 1, we compare our result with the existing works.

<table>
<thead>
<tr>
<th>Event</th>
<th>Required Sample Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximov [8]</td>
<td>( Z_{2k+1}^{(2)} = Z_{2k+2}^{(2)} )</td>
</tr>
<tr>
<td>Tsunoo et al. [15]</td>
<td>( Z_1^{(1)} = Z_2^{(1)} )</td>
</tr>
<tr>
<td>Our</td>
<td>( Z_1^{(2)} = 2 )</td>
</tr>
</tbody>
</table>

**Table 1** Comparison of our work on RC4A with existing results.

Experiments with 1 billion random keys demonstrate that \( P(Z_1^{(2)} = 2) = 0.003925 \) (\( \approx \frac{1}{N} + \frac{1}{N^2} \)).
Further Non-randomness in RC4, RC4A and VMPC

VMPC PRGA

\[ j = S[j + S[i]] \]
\[ Z = S[S[j] + 1] \]
Swap \( S[i] \leftrightarrow S[j] \)
\[ i = i + 1 \]

Fig. 4 Pseudo-Random Generation Algorithm of VMPC.

Fig. 5 Before and after swap when \( i = 1 \). Here second keystream is \( Z_1 = Z \).

4 New bias on VMPC

VMPC is another variant of RC4 proposed in FSE 2004 [16]. The key generation algorithm of VMPC is as shown in Fig. 4.

We denote the index by \( j_t \), the state by \( S_t \) and the output keystream by \( Z_t \) when \( i = t \). Hence the first output keystream will be \( Z_0 \). VMPC byte generation algorithm is presented pictorially in Figure 5.

We start with the following lemma.

**Lemma 3** Let \( j_2, j_3 \) be the values of \( j \) when \( i = 2, 3 \) respectively. Then

\[ P(j_2 = 1 | j_3 = 2) \approx \frac{2}{N} - \frac{1}{N^2}. \]

**Proof** We have \( j_3 = S_3[j_2 + S_3[3]] \), where \( S_3 \) is the updated state array when \( i = 3 \). Note that if \( j_2 = 1 \) and \( S_3[3] = 2 \), \( j_3 \) will always be 2. Hence

\[ P(j_2 = 1 \land j_3 = 2 | S_3[3] = 2) = P(j_2 = 1 | S_3[3] = 2) = \frac{1}{N}. \]

Again if \( j_2 = 1 \) and \( S_3[3] \neq 2 \), \( j_3 \) can be 2 due to random association. Hence

\[ P(j_2 = 1 \land j_3 = 2) = P(j_2 = 1 \land j_3 = 2 | S_3[3] = 2)P(S_3[3] = 2) + P(j_2 = 1 \land j_3 = 2 | S_3[3] \neq 2)P(S_3[3] \neq 2) \\
= \frac{1}{N} \cdot \frac{1}{N} + \frac{1}{N} \cdot \frac{1}{N} \cdot \left(1 - \frac{1}{N}\right) \\
= \frac{2}{N^2} - \frac{1}{N^3}. \]

Assuming \( P(j_3 = 2) = \frac{1}{N} \), \( P(j_2 = 1 | j_3 = 2) = \frac{P(j_2 = 1 \land j_3 = 2)}{P(j_3 = 2)} = \frac{2}{N} - \frac{1}{N^2}. \) □
Now we will prove the following distinguishing attack on VMPC.

**Theorem 4** Assume initial permutation $S$ are taken uniformly from the set of all possible permutation of the set $\{0, 1, \ldots, N - 1\}$. Also $j$ is taken uniform at random from the set $\{0, 1, \ldots, N - 1\}$. Then the probability $P(Z_1 = Z_3) \approx \frac{1}{N} + \frac{9}{N^2}$, where $Z_1, Z_3$ are keystreams when $i = 1$ and 3 respectively.

**Proof** $P(Z_1 = Z_3 \mid j_3 = 1)$: First we will calculate $P(Z_1 = Z_3 \mid j_3 = 1)$. Consider the event

$$E = \left\{ S_1[S_1[j_1]] + 1 \not\in \{j_1, j_2\}, S_1[S_1[j_1]] + 1 \not\in \{1, 2\}, S_1[j_1] \not\in \{j_1, j_2\}, S_1[j_1] \not\in \{1, 2\} \right\}.$$ 

Now consider the event $E' = E \land (j_2 \neq 1)$.

If $E'$ holds together with $j_3 = 1$, then $Z_3$ will always be equal to $Z_1$. In $E'$, there are 9 conditions all together. Hence $P(E') \approx \frac{9}{N}$. Without above conditions, still $Z_3$ can be equal to $Z_1$ due to random association, probability in this path will be $\frac{1}{N}$. Hence

$$P(Z_1 = Z_3 \mid j_3 = 1) = P(Z_1 = Z_3 \mid j_3 = 1, E')P(E') +$$

$$P(Z_1 = Z_3 \mid j_3 = 1, E^c)P(E^c)$$

$$\approx 1 \cdot \left(\frac{9}{N}\right) + \frac{1}{N} \cdot \frac{9}{N}$$

$$= 1 - \frac{9}{N} + \frac{9}{N^2} = p_1.$$ 

$P(Z_1 = Z_3 \mid j_3 = 2)$: Now we will calculate $P(Z_1 = Z_3 \mid j_3 = 2)$.

We have

$$P(Z_1 = Z_3 \land j_3 = 2) = P(Z_1 = Z_3 \land j_3 = 2 \land E \land j_2 = 1)$$

$$+ P(Z_1 = Z_3 \land j_3 = 2 \land (E^c \cup j_2 \neq 1))$$

Now

$$P(Z_1 = Z_3 \land j_3 = 2 \land E \land j_2 = 1) = P(Z_1 = Z_3 \mid j_3 = 2 \land E \land j_2 = 1).$$

$$P(j_3 = 2 \land E \land j_2 = 1) \approx P(Z_1 = Z_3 \mid j_3 = 2 \land E \land j_2 = 1) \cdot P(E) \cdot P(j_3 = 2 \land j_2 = 1)$$

$$= P(Z_1 = Z_3 \mid j_3 = 2 \land E \land j_2 = 1) \cdot P(E) \cdot P(j_2 = 1 \mid j_3 = 2) \cdot P(j_3 = 2)$$

If the event $E$ holds with $j_3 = 2$ and $j_2 = 1$, then $Z_3$ will be always equal to $Z_1$. Also the event $E$ holds with probability $1 - \frac{8}{N}$. Again from Lemma 3, we know that probability of $j_2 = 1$ given $j_3 = 2$ is $\frac{2}{N} - \frac{1}{N^2}$. Hence

$$P(Z_1 = Z_3 \land j_3 = 2 \land E \land j_2 = 1) = 1 \cdot \left(1 - \frac{8}{N}\right) \cdot \left(\frac{2}{N} - \frac{1}{N^2}\right) \cdot \frac{1}{N}.$$
Now if the event $E$ holds with $j_3 = 2$ and $j_2 \neq 1$, $Z_3$ will be always different to $Z_1$. Hence $P(Z_1 = Z_3 \land j_3 = 2 \land E^c) = P(Z_1 = Z_3) \cdot P(j_3 = 2) \cdot P(E^c)$.

Hence,

$$P(Z_1 = Z_3 \land j_3 = 2) = \left(1 - \frac{8}{N}\right) \cdot \left(\frac{2}{N} - \frac{1}{N^2}\right) \cdot \frac{1}{N} + \frac{8}{N^2}.$$

Finally we get

$$P(Z_1 = Z_3 | j_3 = 2) = \left(1 - \frac{8}{N}\right) \left(\frac{2}{N} - \frac{1}{N^2}\right) + \frac{8}{N^2} = p_2.$$

$P(Z_1 = Z_3 | j_3 \neq 1, 2)$: Now consider the case when $j_3 \notin \{1, 2\}$. So, We are interested to calculate $P(Z_1 = Z_3 | j_3 = x)$, where $x \notin \{1, 2\}$. By tracing the state, one can note that when the event $E$ holds, $Z_1$ can not be $Z_3$ given $j_3 = x$. Hence we have

$$P(Z_1 = Z_3 | j_3 = x) = P(E^c) \cdot P(Z_1 = Z_3 | j_3 = x, E^c)$$

$$= P(E^c) \cdot \frac{1}{N}$$

$$= \frac{8}{N} \cdot \frac{1}{N}$$

$$= \frac{8}{N^2}$$

$$= p_3.$$

Hence

$$P(Z_1 = Z_3) = P(Z_1 = Z_3 | j_3 = 1)P(j_3 = 1) + P(Z_1 = Z_3 | j_3 = 2)P(j_3 = 2)$$

$$+ \sum_{x \notin \{1, 2\}}^{N-1} P(Z_1 = Z_3 | j_3 = x)P(j_3 = x) = \frac{p_1}{N} + \frac{p_2}{N} + p_3 \left(1 - \frac{2}{N}\right).$$

Now putting the values of $p_1, p_2$ and $p_3$ in the above expression, we have

$$P(Z_1 = Z_3) = \frac{1}{N} + \frac{1}{N^2} - \frac{16}{N^3} + \frac{8}{N^4} \approx \frac{1}{N} + \frac{1}{N^2}.$$  

Hence with a sample of size $N^3$, one can mount a distinguishing attack on VMPC. So for $N = 256$, with a sample of size $2^{24}$ one can distinguish VMPC from a random number generator. The best distinguisher for this cipher is due to [8] with sample size $2^{22}$.

Using exactly the same approach as Theorem 4, we have the most significant long term bias on VMPC in the following lemma.

**Conjecture 1** Assume that the initial permutation $S$ is taken uniformly from the set of all possible permutations of the set $\{0, 1, \ldots, N-1\}$. Also $j$ is taken uniformly at random from the set $\{0, 1, \ldots, N-1\}$. Then the probability $P(Z_{kN+1} = Z_{kN+3}) \approx \frac{1}{N} + \frac{1}{N^2}$, where $Z_l$ is the $l+1^{th}$ keystream byte and $k$ is a non-negative integer.

Experiments with 1 billion sample size show that $P(Z_{kN+1} = Z_{kN+3}) = 0.003921$ ($\approx \frac{1}{N} + \frac{1}{N^2}$).
5 Conclusion

In this paper we have studied new distinguishers of RC4 and its variants RC4A and VMPC. We have proved that the second output byte of RC4 has negative bias of $\frac{3}{N^2}$ towards 2. This is second highest distinguisher after the work of [7]. Distinguisher on RC4A in our approach needs $N^3$ sample size, same as the existing work [15]. In the case of VMPC, we have proved that the fourth output byte has a positive $\frac{1}{N^2}$ bias towards the second output. This can be easily converted to a long term bias with same complexity. This is the strongest distinguisher against VMPC. The required sample size in our case is $N^3$, whereas the existing best attack [8] requires $N^4$ samples.

We have observed in RC4 that $P(Z_2 = N^2 + 1) = \frac{1}{N^2} - 2 \frac{1}{N^3}$. In [6], authors proved $r^{th}$ output byte $Z_r$ in RC4 has positive bias towards 0 for $3 \leq r \leq N - 1$. We have observed that $Z_r$ also has positive bias towards r for $4 \leq r \leq \frac{N}{2}$. In the full version of the paper, we will include these results.

Acknowledgment: The author would like to thank Prof. Subhamoy Maitra and the anonymous WCC reviewers for their detailed comments on the technical issues of this paper.

References

Further Non-randomness in RC4, RC4A and VMPC

On the Four-Round AES Characteristics

Vincent Rijmen, Deniz Toz, Kerem Varıcı

Abstract In this work we study the AES characteristic. The main aim of this work is to compute the fixed-key probabilities of four-round AES characteristics. We use plateau characteristics and follow the approach that was used to compute two-round AES characteristics. However, it is not possible to use the theory immediately for the four rounds. Therefore, we first solve some of the problems to make computation feasible for four rounds. Then, we revisit the plateau theorem and generalize it.

1 Introduction

Fixed-key probability of characteristics in block ciphers is a topic that has been studied earlier for Feistel structures and AES-like block ciphers [2, 4, 10]. It was shown that the probabilities are dependent on the value of the key. Furthermore, in [1, 12], all fixed-key probabilities for two round characteristics of AES were computed. Their results show that the dependency on the value of the key is even structured in AES for two rounds and such characteristics are called plateau characteristics.

The extension of knowledge on the AES characteristics continued with [6]. In this paper, four round characteristics were investigated and bounds were given for their expected differential probabilities (EDP). In addition, some estimations of the maximum expected differential probability (MEDP) for four rounds of AES were provided in [12, 15]. However, there exists no result for the fixed-key probabilities of characteristics for four rounds of AES. Even though these results are only for a reduced number of rounds (four out of ten) for AES they are relevant. There exist some algorithms like the message authentication code Pelican [8] and the stream cipher LEX [3] that use four AES-round as core operation.

This work was sponsored by the Research Fund K.U.Leuven, OT/08/027

V. Rijmen · D. Toz · K. Varıcı
KU Leuven Department of Electronical Engineering ESAT SCD-COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
E-mail: {vincent.rijmen, deniz.toz, kerem.varici}@esat.kuleuven.be
Our contribution: Our work aims to extend the results on the fixed-key probability of characteristics for AES-like structures from two to four rounds. First, we present results on the propagation of affine spaces through nonlinear maps. Then, we revisit the two-round plateau characteristic theorem [10] and generalize it for multiple keys. Next, we focus on the distribution of secondary differences for two rounds of AES. After defining the mega-boxes for four rounds of AES, we further investigate activity patterns and the characteristics with 25 active S-boxes. Finally, we use all the properties that we derive to compute the height (the number of right pairs for a characteristic in binary logarithm) for four rounds of AES.

This paper is organized as follows: In Sec. 2, we give a brief summary of previous work and introduce the required definitions. Then we define two theorems in Sec. 3 and Sec. 4 to improve the existing work on two rounds of AES. Later, we present the results on secondary differences in Sec. 5 which is followed by the results on four rounds of AES in Sec. 6 and Sec. 7. Finally, we conclude the paper by explaining possible future works in Sec. 8.

2 Background, Notations and Definitions

We consider invertible maps on inputs of size \( n \) bits. A differential over a map \( \gamma \) is denoted by \((a, b)\) where \( a \) is the input difference and \( b \) is the output difference.

Let \( F_{(a, b)} \), \( G_{(a, b)} \) be the sets that contain the input values, respectively the output values, for the right pairs of the differential \((a, b)\).

\[
\begin{align*}
F_{(a, b)} &= \{ x | \gamma(x) + \gamma(x + a) = b \} \\
G_{(a, b)} &= \gamma(F_{(a, b)})
\end{align*}
\]

A differential \((a, b)\) is called a planar differential, if \( F_{(a, b)} \) and \( G_{(a, b)} \) form affine subspaces [10]. In that case, we can write:

\[
\begin{align*}
F_{(a, b)} &= p + U_{(a, b)} \\
G_{(a, b)} &= q + V_{(a, b)}
\end{align*}
\]

where \( U_{(a, b)} \) and \( V_{(a, b)} \) are uniquely defined vector spaces, \( p \) any element in \( F_{(a, b)} \) and \( q \) any element in \( G_{(a, b)} \). Note that if a differential \((a, b)\) has exactly two or four right pairs, then it is always planar [10]. If all differentials through a map are planar, then the map is called a planar map.

Obviously, the sets \( F_{(a, b)}, G_{(a, b)}, U_{(a, b)}, V_{(a, b)} \) have the same cardinality. This cardinality divided by \( 2^{n_b} \) is called the probability of the differential, and denoted by \( \text{DP}(a, b) \). If \( \text{DP}(a, b) = 2^{2^{n_b}} \), then \( U_{(a, b)} = \{0, a\} \) and \( V_{(a, b)} = \{0, b\} \). If \( \text{DP}(a, b) = 4^{2^{n_b}} \), then there are some \( \alpha, \beta \) such that \( U_{(a, b)} = \{0, \alpha, \alpha + a\} \) and \( V_{(a, b)} = \{0, b, \beta, \beta + b\} \). We call \( a, b \) the primary differences and \( \alpha, \beta \) the secondary differences.

A composed map consists of a sequence of sub-maps. A characteristic \( Q \) through a composed map is a sequence of differences \( a, b, c, \ldots \). The sequence consists of an input difference \( a \), followed by the output differences of all the steps of the composed map. A characteristic over a composed map has a differential probability \( \text{DP}(Q) \). We define \( F_Q, G_Q \) as in Equation 1, and \( U_Q, V_Q \) as in Equation 2 if they exist.
For a keyed map, we can define differential probabilities DP\(_k(a,b)\) and DP\(_k(Q)\) for each value \(k\) of the key. Then, the expected differential probability (EDP) is the average of the differential probability over all keys. We denote by \(K(a,b)\), respectively \(K_Q\), the set of keys for which DP\(_k(a,b) > 0\) or DP\(_k(Q) > 0\). The weight \(weight(Q)\) of a possible differential or a characteristic is minus the binary logarithm of their EDP. A special case is plateau characteristics, where the dependency of the DP\(_k\) on the value of the key is very structured and defined as follows:

**Definition 1 (Plateau characteristic [10])** A characteristic \(Q\) is a plateau characteristic with height \(height(Q)\) if and only if:

1. For a fraction \(2^{n_b - (weight(Q) + height(Q))}\) of the keys \(DP[k](Q) = 2^{height(Q) - n_b}\).
2. For all other keys \(DP[k](Q) = 0\).

The planar property of differentials is extended to characteristics as follows:

**Definition 2** A characteristic \(Q\) is input-planar (respectively output-planar) if for all values of the key it holds that \(F_Q[k]\) (respectively \(G_Q[k]\)) is either empty or an affine subspace.

### 3 Propagation of affine spaces

By definition, the action of an affine map on an affine subspace produces a new affine subspace. For a nonlinear map, this is in general not the case. In this section, we investigate some special, invertible, non-linear maps that can carry affine subspaces to affine subspaces under some sufficient conditions. Gathering these sufficient conditions together is important in the case of block ciphers (in our case for AES-like block ciphers) to work with plateau characteristics. The non-linear map used in AES is a subset of the mappings that is defined in Theorem 1, therefore computation of right pairs for a chosen characteristic through plateau characteristics, i.e. \(height(Q)\) becomes possible for a subset of four round characteristics which is covered by Theorem 1. But before that, some required definitions will be given:

**Definition 3 (Bricklayer map [7])** A map \(\gamma\) is a bricklayer map if it can be decomposed into a number of boolean functions that operate independently on coordinates of the input vector i.e. subsets of the input vector’s bits.

A bricklayer map can be considered as multiple maps (S-boxes) that operate on smaller inputs, applied in parallel. In what follows, we will consider bricklayer maps operating on inputs of \(m\) bricks of \(n = n_b/m\) bits each.

\[ b = \gamma(a) \iff (b_1, b_2, \ldots, b_m) = (\gamma_1(a_1), \gamma_2(a_2), \ldots, \gamma_m(a_m)) \]

Higher-order differences are defined recursively in [13]. We use here a slightly different, but equivalent formulation.

**Definition 4 (Second order difference)** We define the second order difference of the map \(\gamma\) over the 2-dimensional vector space \(U = \{0, u_1, u_2, a_1 + u_2\}\) as:

\[ \Delta^2_U \gamma(x) = \gamma(x) + \gamma(x + u_1) + \gamma(x + u_2) + \gamma(x + u_1 + u_2). \]
Theorem 1  Let $\gamma$ be an invertible map. Let $U$ be a subspace of $\mathbb{F}^{nm}_2$ and $p \in \mathbb{F}^{nm}_2$. Then $F = U + p$ is an affine subspace of $\mathbb{F}^{nm}_2$. The set $G = \gamma(F)$ is an affine subspace of $\mathbb{F}^{nm}_2$ when one of the following conditions holds:

1. $\dim(U) = 1$
2. $\dim(U) = 2$ and the second order differential of $\gamma$ in $p$ equals 0:
\[ \Delta^2_U \gamma(p) = 0 \] (3)
3. $\gamma$ is a bricklayer map, $U \subseteq U_1 \times U_2 \times \cdots \times U_m$ with $\dim(U_i) \leq 2$, $\forall i$ and $\Delta^2_{U_i} \gamma(p) = 0$ for all $i$ where $\dim(U_i) = 2$.

For the proof of this theorem, we refer to the full version of the paper. Note that (3) is always satisfied if $\{p, p + u_1\}$ and $\{p + u_2, p + u_1 + u_2\}$ are two right pairs of a differential $(u_1, \beta)$ for any $\beta$.

Theorem 1 is applicable to all AES-like constructions where the underlying S-box is similar to the one used in AES (i.e., the difference distribution table (DDT) of the AES S-box contains either zero, two or four input pairs mapped to one of the output differences, hence, $\dim(U_i) \leq 2$).

4 The plateau theorem revisited

Let $\sigma[k]$ denote a simple key addition: $\sigma[k](x) = k + x$. Let $\rho_0[k]$ be a map composed of three invertible sub-maps, see Figure 1:
\[ \rho_0[k] = \alpha_2 \circ \sigma[k] \circ \alpha_1, \]
where the maps $\alpha_1, \alpha_2$ are planar maps. A characteristic through $\rho_0$ consists of a differential over $\alpha_1$, followed by a differential over $\sigma[k]$, followed by a differential over $\alpha_2$. A differential over $\sigma[k]$ has $\text{DP} > 0$ if and only if its output difference equals its input difference. Hence, a characteristic with $\text{DP} > 0$ through $\rho_0$ is completely characterized by three differences $Q = (a, b, c)$, where $(a, b)$ is the differential over $\alpha_1$, $(b, b)$ is the differential over $\sigma[k]$ and $(b, c)$ the differential over $\alpha_2$.

The two-round plateau characteristic theorem [10] states that all characteristics through $\rho_0$ are plateau characteristics. Furthermore, it was proven that $K_Q = G_{(a,b)} + F_{(b,c)}$. We will now generalize the plateau characteristic theorem by considering maps of the following form:
\[ \rho_1(k, k_1, k_2) = \alpha_2[k_2] \circ \sigma[k] \circ \alpha_1[k_1], \]
where the maps $\alpha_1[k_1], \alpha_2[k_2]$ are arbitrary key-dependent maps and the keys $k, k_1, k_2$ are independent. Secondly, we will look at the structure of the sets of right pairs of characteristics through $\rho_1$. Reasoning in the same way as above, we can state that a characteristic $Q$ through $\rho_1$ is the concatenation of two characteristics: $Q = Q_1 | Q_2$, where $Q_1$ is a characteristic through $\alpha_1$ and $Q_2$ is a characteristic through $\alpha_2$, see also Figure 2. Now, our aim is to investigate the structure of the right pairs under the new mapping $\rho_1$. 

4
4.1 Number of right pairs

Let \( Q \) be a characteristic over \( \rho_1 \). Using the notation defined above, we have:

**Theorem 2** If \( Q_1 \) is output-planar and \( Q_2 \) is input-planar, then for all values of \( k, k_1, k_2 \), \( \text{DP}[k, k_1, k_2](Q) \) is an integer power of 2.

**Proof** The proof uses similar arguments as the proof of the plateau characteristic theorem [10]. Denote by \( H_Q[k, k_1, k_2] \) the set \( \alpha_1(F_Q[k, k_1, k_2]) \). By definition, we have that:

\[
H_Q[k, k_1, k_2] = G_{Q_1}[k_1] \cap (F_{Q_2}[k_2] + k). \tag{4}
\]

\( H_Q[k, k_1, k_2] \) is non-empty if and only if \( \exists p \in F_{Q_2}[k_2], \exists q \in G_{Q_1}[k_1] \) such that \( k = p + q \). It follows that

\[
(k, k_1, k_2) \in K_Q \Leftrightarrow H_Q[k, k_1, k_2] \neq \emptyset \Leftrightarrow k \in G_{Q_1}[k_1] + F_{Q_2}[k_2]. \tag{5}
\]

If \( (k, k_1, k_2) \in K_Q \), then translation of (4) by \( q \) gives:

\[
H_Q[k, k_1, k_2] + q = (G_{Q_1}[k_1] + q) \cap (F_{Q_2}[k_2] + p). 
\]
We are now ready to state how varying the key $k$ influences the values of the right pairs of $Q$. The proof of this corollary not given in this paper due to the page limitations.

### Corollary 1
For any $k, k_1, k_2$,

1. The set $\{H_Q[k, k_1, k_2] | k \in G_{Q_1}[k_1] + F_{Q_2}[k_2]\}$ forms a partition of $G_{Q_1}$.
2. The set $\{F_Q[k, k_1, k_2] | k \in G_{Q_1}[k_1] + F_{Q_2}[k_2]\}$ forms a partition of $F_Q$.
3. The set $\{H_Q[k] + k | k \in G_{Q_1}[k_1] + F_{Q_2}[k_2]\}$ forms a partition of $F_Q$.
4. The set $\{G_Q[k, k_1, k_2] | k \in G_{Q_1}[k_1] + F_{Q_2}[k_2]\}$ forms a partition of $G_{Q_1}$.
5 Two rounds of AES

Several ciphers [5,16] that use S-boxes and linear transformations can be described using the structure of a super box:

**Definition 5 (Super box [10])** A super box maps an array of \( m \) elements \( a_i \) to an array \( e \) of \( m \) elements \( e_i \). Each of the elements has size \( n \). A super box takes a key \( k \) of size \( m \times n = nb \). It consists of the sequence of four transformations (layers): Substitution, Mixing, Round Key Addition, Substitution.

Two rounds of AES can be described as four parallel instances of the AES super box [7]. So, instead of dealing with the classical 8-bit AES S-boxes, one can consider 32-bit S-boxes each composed of two AES S-box layers surrounding one MixColumns (MC) and one AddRoundKey (AK) function (See top left corner of Figure 4). A differential characteristic through the AES super box consists of a sequence of four differences: the input difference \( a \), the difference after the first substitution \( b \), the difference after the mixing step which is equal to the difference after the round key addition \( d \), and the output difference after the second substitution \( e \). These characteristics are denoted by \( Q = (a, b, d, e) \).

Secondary differences play an important role in the analysis. In order to describe the right pairs as vector spaces, we need to know the value of the secondary differences. While for S-boxes finding this value is quite easy, for the case of super boxes the situation is more complicated.

Let \( Q = (a, b, d, e) \) be a plateau characteristic with height 2 through the AES super box. Then we know that for each value of the key \( F_Q \) and \( G_Q \) are empty or they are affine spaces of dimension 2. The same holds for \( H_Q = \text{SubBytes}(F_Q) = \text{SB}(F_Q) \) and \( \text{SB}^{-1}(G_Q) \). Which values are in the affine spaces, depends on the value of the key. From the proof of the plateau characteristic theorem, however, we know that the vectors spanning the subspace parallel to \( H_Q \) do not depend on the value of the key. One of the vectors is \( b \), the difference specified by the characteristic. We call the second vector a secondary difference and denote it by \( \beta \). Similarly, the affine space \( \text{SB}^{-1}(G_Q) \) is parallel to the subspace spanned by \( d \) and \( \delta \) and \( \delta = \text{MC}(\beta) \). The subspace parallel to \( G_Q \) always contains the vector \( e \). We denote the second vector of the subspace by \( \epsilon \). This vector depends on the value of the key (See Figure 3).

![Fig. 3 Effect of the key on secondary differences](image)

**Theorem 3** Let \( Q = (a, b, d, e) \) be a characteristic of height 2 (through the AES super box) and let \( \delta = (\delta_0, \delta_1, \delta_2, \delta_3) \) be a secondary difference for \( d \). Then, each secondary difference for \( e \), denoted by \( \epsilon = (\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3) \) with \( \text{DP}(\delta_i, \epsilon_i) > 0 \) where \( i \in \{0, 1, 2, 3\} \)
is observed for

\[ 2^{\dim(V_{(a,d)}+U_{(d,e)})} \times \prod_{\{i | d_i > 0\}} \text{DP}(\delta_i, \epsilon_i) \]

values of the key.

**Proof** We know that there are \(|K_Q| = 2^{\dim(V_{(a,d)}+U_{(d,e)})}\) keys for which \(\text{DP}(Q) > 0\).

Hence, this is the maximum number that any secondary difference can occur.

Moreover, since the height of the characteristic is two, there will be four elements (not necessarily different) in each of the active and passive bricks. All the elements in the active part of the difference \(d\) can be transformed to the \(e\) directly since the differences \(d\) and \(e\) fixed. For each of the passive bricks, let us denote the projections of four elements in \(SB^{-1}(G_Q)\) to \(i\)-th S-box by \(\{p_{d_i}, p_{d_i}, q_{d_i}, q_{d_i}\}\) where \(p_{d_i} = p_{d_i}^*\) and \(q_{d_i} = q_{d_i}^*\). Then, the primary difference equals to \(p_{d_i} \oplus p_{d_i}^* = 0\) and the secondary difference is \(q_{d_i} \oplus q_{d_i}^* = \delta_i \neq 0\). The corresponding projections in \(G_Q\) can be called as \(\{p_{e_i}, p_{e_i}, q_{e_i}, q_{e_i}\}\) and their values depend on the key value. The primary difference again equals to \(p_{e_i} \oplus p_{e_i}^* = 0\) and the secondary difference is \(q_{e_i} \oplus q_{e_i}^* = \epsilon_i \neq 0\). Since, a non-linear substitution operation is used in the computation of \(e\), the secondary difference \(p_{e_i} \oplus q_{e_i}\) can occur if \(\text{DP}(p_{e_i} \oplus q_{e_i}, p_{e_i} \oplus q_{e_i}) > 0\).

Thus, the number of secondary differences and their distribution are directly effected by \(\text{DP}(\delta, \epsilon)\) and each secondary difference in \(e\) occurs for

\[ 2^{\dim(V_{(a,d)}+U_{(d,e)})} \times \prod_{\{i | d_i > 0\}} \text{DP}(\delta_i, \epsilon_i) \]

values of the keys.

**Corollary 2** Let \(Q = (a,b,d,e)\) be a characteristic of height 2 (through the AES super box). A secondary difference \(\epsilon\) occurs for some value of the key if and only if \(G_{(d,e)} \cap G_{(\delta,\epsilon)} \neq \emptyset\).

**Proof** We know from Corollary 1 that all elements of \(G_{(d,e)}\) will be present in \(G_Q[k]\) for some value of the key \(k\). Hence, if \(G_{(d,e)} \cap G_{(\delta,\epsilon)} \neq \emptyset\), then \(\exists k, y : y \in G_Q[k] \cap G_{(\delta,\epsilon)}\). Since \(y \in G_{(d,e)}\), also \(y + \epsilon \in G_{(d,e)}\). Since \(y \in G_Q[k]\), we have \(SB^{-1}(y), SB^{-1}(y) + d, SB^{-1}(y) + d + \delta, SB^{-1}(y) + d + \delta \subseteq SB^{-1}(G_Q[k])\). Hence \(\exists k\) for which \(y, y + \epsilon, y + \epsilon, y + \epsilon = \epsilon \in G_Q[k]\).

On the other hand, we know that \(G_Q \subseteq G_{(d,e)}\) for all values of the key. Hence, if \(G_{(d,e)} \cap G_{(\delta,\epsilon)} = \emptyset\), then \(\nexists y, k\) for which \(y, y + \epsilon, y + \epsilon, y + \epsilon = \epsilon \in G_Q[k]\). \(\square\)

6 Four rounds of AES

6.1 Introduction to Mega-Boxes

Similar to two rounds of AES, four rounds of AES can be described as a mega-box, where the elements are 32-bit words and the AES super boxes defined above are equivalent to (key-dependent) S-boxes. The linear transformation in the middle is a combination of ShiftRows, MixColumns and ShiftRows respectively. We will refer to this linear transformation as SMS for the rest of the text. Figure 4 illustrates this.
It can be shown that SMS is a map whose branch number is 5. Therefore, a characteristic over such a mega-box consists of 5 to 8 sub-characteristics, each over an AES super box. We denote the characteristics over the first and the second layer of super boxes by \((a, b, d, e)\) and \((f, g, i, j)\), respectively.

![Fig. 4 Representation of four rounds of AES](image)

**Fig. 4** Representation of four rounds of AES

**Fig. 5** Representation of SMS layer and super boxes

6.2 Four-round characteristics with 25 active S-boxes

Four-round characteristics have at least 25 active S-boxes. Determining the number of characteristics with exactly 25 S-boxes is considerably easier than the general case, and that is why we consider this special case in this paper.

Firstly, observe that the SMS map consists of 4 identical maps acting in parallel. We will denote these maps here by \(SMS_l\), with \(l = 1, 2, 3, 4\) (See Figure 5). Each \(SMS_l\) takes one byte from each super box as input or output, and is in fact an MDS map. By examining the SMS, one can make the following observations:

**Observation 1** Let \(m_i\) be the number of active bytes at the input and \(n_i\) be the number of active bytes at the output of \(SMS_l\). Then, the total number of active super boxes is at least \(\max_i\{m_i\} + \max_i\{n_i\}\).

It can be seen that the number of active super boxes in a 4-round characteristic is determined by the number of active positions in the \(SMS_l\) maps.

**Definition 6 (Activity pattern [9])** The activity pattern \((AP)\) has a single bit for each S-box position indicating whether it is passive (0) or active (1).

**Theorem 4** A four-round characteristic has exactly 5 active super boxes if and only if each of the \(SMS_l\) has either 5 active bytes with the same number of active bytes at inputs/outputs where the activity patterns of the \(SMS_l\) maps at the input/output are related by a rotation or they have no active inputs/outputs. By this, we mean that if \(SMS_1\) has the activity pattern \((1100; 1110)\), then \(SMS_2\) has either \((1001; 0111)\) or \((0000; 0000)\). Similarly \(SMS_3\) and \(SMS_4\) have the activity patterns \((0011; 1011)\) and \((0110; 1101)\) respectively or all zeroes. For \(k > l\) the equations are given as:

\[
AP(SMS_k) = AP(SMS_l) \ll (k - l) \quad \text{at the input} \quad (6)
\]

\[
AP(SMS_k) = AP(SMS_l) \gg (k - l) \quad \text{at the output} \quad (7)
\]
Proof When $A_1$ active bytes results in $A_2$ active bytes after MDS operation, we denote it by $A_1 \rightarrow A_2$. Here, we show only the proof of the first two cases out of four, since the proof of other two cases are similar to the second one.

$\Rightarrow$ Assume that a four-round characteristic has exactly five active super boxes. Then:

case 1. Assume that only one of the $\text{SMS}_l$ is active with $A$ active bytes. Then it has $m \rightarrow (A - m)$ for some $m \in \{1, 2, 3, 4\}$. By the Observation 1, $m$ of the super boxes at the input and $A - m$ of the super boxes at the output of $\text{SMS}$ will be active and each of them has exactly one byte entering the MDS. Then, we have

$$m + (A - m) = A$$

active super boxes, and hence $A = 5$.

case 2. Two of the $\text{SMS}_l$ maps are active with $m_1 \rightarrow (A_1 - m_1)$ and $m_2 \rightarrow (A_2 - m_2)$. Without loss of generality assume that $m_1 < m_2$. Then, by Observation 1, at least $\max\{m_1, m_2\} = m_2$ of the super boxes at the input and $\max\{A_1 - m_1, A_2 - m_2\}$ of the super boxes at the output have to be active. Then

- If $A_1 - m_1 \geq A_2 - m_2$ then the total number of active super boxes is $m_2 + (A_1 - m_1) = A_1 + (m_2 - m_1)$. We know that $A_1 \geq 5$ and $m_2 - m_1 \geq 0$. Then in order to have 5 active super boxes $A_1 = 5$ and $m_1 = m_2$. It follows that $A_1 - m_1 \geq A_2 - m_2 \iff A_1 \geq A_2$ and since $A_2 \geq 5$, we obtain $A_2 = 5$.

- If $A_2 - m_2 \geq A_1 - m_1$ then the total number of super boxes is $m_2 + (A_2 - m_2) = A_2$, and hence $A_2 = 5$. Since $A_2 - A_1 \geq m_2 - m_1 \geq 0$ and $A_1 \geq 5$, we obtain $A_2 = 5$.

As a result, we deduce that if a four round characteristic has exactly five active super boxes then each $\text{SMS}_l$ is either passive or active with five active S-boxes. Moreover the active $\text{SMS}_l$ have the same number of active bytes at the inputs and outputs.

$\Leftarrow$ Assume that each $\text{SMS}_l$ has either five active bytes or has no active inputs/outputs bytes. Since the $\text{SMS}$ layer also satisfies the equations 6 and 7, each $\text{SMS}_l$ must be in the same form $n \rightarrow (5 - n)$. Then, according to Observation 1, $\max_1(n_l) + \max_1(5 - n_l) = \max_2(n) + \max_2(5 - n) = 5$ is the number of active super boxes.

\(\square\)

**Corollary 3** A four-round characteristic can have exactly 25 active S-boxes if and only if there are exactly 5 active super boxes and each of the active super boxes has exactly 5 active S-boxes. Moreover each of the active super boxes at the input (output) have the same pattern (i.e. if one of the super boxes has $n$ active bytes in the input and $(5 - n)$ active bytes in the output, then the other super boxes at the same level (input/output of $\text{SMS}$) also have the same number of active bytes in the input and output).

*Proof* It is obvious that if there are exactly 5 super boxes active and each of the active super boxes has exactly 5 active S-boxes then there are exactly 25 active S-boxes. So, assume that a four-round characteristic has exactly 25 active S-boxes, and let the cases be as in Theorem 4. Then, for each case, we know that $m$ of the super boxes at the input and $5 - m$ of the super boxes at the output of $\text{SMS}$ are
active. Since each SMS is an MDS with branch number five, this is the minimum number of active super boxes that one can obtain. We also know that if a super box is active, then it has at least 5 active S-boxes. Then the result follows. Moreover, let n be the number of active SMS’s. Then, each super box has exactly n active S-boxes before the MDS operation. Therefore, the resulting patterns are \((5 - n) \rightarrow n\) for the active super boxes at the input and \(n \rightarrow (5 - n)\) for the active super boxes at the output of SMS respectively.

\[\square\]

7 Height computation for four rounds of AES

In this section, we compute the four rounds AES characteristics with an approach similar to [10, 14]. We replace the S-box operation by a super box and the MDS operation by the SMS layer. Then, we use the plateau characteristics through mega-boxes.

7.1 An example for four rounds of AES with \(h \geq 2\)

We present the effect of secondary differences on the height with an example. Since, the super box is a key dependent operation, secondary differences change for each possible key value. We first fix a differential characteristic given in Figure 6.

![Fig. 6](image)

A four-round characteristic with multiple heights (differences are in hexadecimal).

Each of the active super boxes has one passive byte at the input and at the output of the SMS layer. Then, there exist 127 possible secondary differences for each of the active super boxes depending on the key value by Theorem 3. As a result, we observe that the height of the four round characteristic ranges between two and four. Independent from the value of the key, the secondary differences in \(d\) are 0x6400DE00, 0x06400DE and 0xDEE006400 for the super boxes at the input of SMS and the secondary differences in \(g\) are 0xA8002E00 and 0x2E00A800 for the super boxes at the output of SMS. Below, we just give one of the 127\(^5\) possible corresponding secondary differences in \(e\) and \(f\) for each height.
7.2 Related differentials over four rounds of AES

In [11], a property of linear maps called related differentials is introduced. Related differentials do not influence the EDP of characteristics, but instead they affect the distribution of their DP[$k$] values.

Definition 7 ([11]) Two differences $x, x^\Diamond$ containing each $m$ elements are related if and only if $x_j x_j^\Diamond (x_j + x_j^\Diamond) = 0$, for $j = 0, 1, 2, \ldots, m - 1$

Definition 8 (Related differentials [11]) Two differentials $(b, c), (b^\Diamond, c^\Diamond)$ for a linear map $M$ are related differentials if and only if $c = M(b), c^\Diamond = M(b^\Diamond)$, the differences $b, b^\Diamond$ are related differences and the differences $c, c^\Diamond$ are related differences.

The four pairs of differences from which all related differentials can be derived by means of rotation and/or multiplication by a scalar are listed in [11, Table 5]. Here, we correct some typos it in and give them in Table 1.

Table 1 The sets of related differentials over MixColumns

<table>
<thead>
<tr>
<th>b</th>
<th>c</th>
<th>b^\Diamond</th>
<th>c^\Diamond</th>
<th>b+b^\Diamond</th>
<th>c+c^\Diamond</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1,4,7</td>
<td>0,9,0,8</td>
<td>5,1,0,7</td>
<td>E,0,D,0</td>
<td>5,0,4,0</td>
<td>E,9,D,B</td>
</tr>
<tr>
<td>0,1,0</td>
<td>9,E,0,0</td>
<td>2,0,1,0</td>
<td>5,1,0,7</td>
<td>2,1,1,3</td>
<td>5,0,4,0</td>
</tr>
<tr>
<td>7,0,7,7</td>
<td>E,9,0,0</td>
<td>7,7,7,0</td>
<td>0,0,E,9</td>
<td>0,7,0,7</td>
<td>E,9,E,9</td>
</tr>
<tr>
<td>0,3,2,0</td>
<td>7,0,7,1</td>
<td>2,0,0,3</td>
<td>7,1,7,0</td>
<td>2,9,2,3</td>
<td>0,1,0,1</td>
</tr>
</tbody>
</table>

In this section, we propose a method to obtain four-round characteristics whose height is at least 2. We know that the height of the four round characteristics depends on the vector spaces $V_{(a,c)}$ and $U_{(f,j)}$. If the difference of each SMS$_l$ is a multiple of one of the related differentials for two-rounds, then these differentials are associated with the same multiple of its related differential (Table 1). Therefore, it is guaranteed that the height is minimum 2, without requiring the use of the secondary differences of the super boxes.

Lemma 2 ($e, f$) is in a set of related differentials over the mixing map SMS, if and only if $(\text{ShiftRows}(e)_l, \text{ShiftRows}^{-1}(f)_l) = (\text{SR}(e)_l, \text{SR}^{-1}(f)_l)$ is in a set of related differentials or equals to zero over MixColumns where $l \in \{0, 1, 2, 3\}$.
Proof Let \((e, f)\) be a differential over \(\text{SMS}\) map. Then, it can be written of the form:

\[
\text{SMS}(e) = f \Rightarrow \text{MC}(\text{SR}(e)) = \text{SR}^{-1}(f).
\]  

If we decompose \(\text{MC}(\text{SR}(e))\) and \(\text{SR}^{-1}(f)\):

\[
(\text{MC}(\text{SR}(e))_1, \text{MC}(\text{SR}(e))_2, \text{MC}(\text{SR}(e))_3) = (\text{SR}^{-1}(f)_0, \text{SR}^{-1}(f)_1, \text{SR}^{-1}(f)_2, \text{SR}^{-1}(f)_3)
\]

Note that if \((\text{SR}(e), \text{SR}^{-1}(f))\) is in a set of related differentials or zero, then we can conclude that \((\text{SR}(e), \text{SR}^{-1}(f))\) is also in a set of related differentials since the second mapping is just a composition of the first one. Let \((\text{SR}(e), \text{SR}^{-1}(f))\) is related with \((\text{SR}(e^*), \text{SR}^{-1}(f^*))\), then \((\text{SR}(e), \text{SR}(e^*)), (\text{SR}^{-1}(f), \text{SR}^{-1}(f^*))\) are related differentials and

\[
\text{SR}(e)_m \cdot \text{SR}(e^*)_m \cdot [\text{SR}(e)_m + \text{SR}(e^*)_m] = 0, \quad m = 0, 1, \ldots, 127,
\]

\[
\text{SR}^{-1}(f)_m \cdot \text{SR}^{-1}(f^*)_m \cdot [\text{SR}^{-1}(f)_m + \text{SR}^{-1}(f^*)_m] = 0, \quad m = 0, 1, \ldots, 127
\]

Note that \(\text{SR}\) (and its inverse \(\text{SR}^{-1}\)) is linear and invertible, then it follows that for each \(m\), at least one of the differences \(e_m, e^*_m, e_m + e^*_m\) equals to zero (for \(f_m, f^*_m, f_m + f^*_m\) as well). Therefore, \((e, e^*)\) and \((f, f^*)\) are also related differentials where \(\text{SMS}(e) = f\) and \(\text{SMS}(e^*) = f^*\). Thus, \((e, f)\) is in a set of related differentials.

\[\square\]

Theorem 5 Let \(Q = Q_1|Q_2\) be a characteristic through a mega box with \(EDP(Q) > 0\) where \(Q_1 = (a, b, d, e)\) is output planar and \(Q_2 = (f, g, i, j)\) is input planar. If \((e, f)\) is in a set of related differentials over the mixing map \(\text{SMS}\), then \(Q\) is a plateau characteristic with \(height(Q) \geq 2\).

Proof Since \(Q_1\) is output planar and \(Q_2\) is input planar, then \(Q\) becomes plateau characteristic by using [10, Thm. 5]. The proof in [11, Thm. 2] can apply here: the S-box operation needs to be changed with the super box operation and mixing step with SMS layer. Then, we need to use Lemma 2 to define related differentials in mega box and the rest of the proof will follow.

8 Conclusion and Further Work

In this work, we show that it is possible to compute fixed-key probabilities of four rounds characteristics for AES under certain conditions by using the plateau characteristics. The best number of right pairs that we found is 16 (i.e. characteristics with probability \(16/2^{128} = 2^{124}\)). These results are open to improvement since the main aim of this work is not concentrated on best results. Instead, this work mainly investigates under which conditions the four round plateau characteristics exists and how they can be computed.

As a future work, it would be interesting to find a way to discover characteristics on four rounds of AES with higher height values. Depending on the results, we can extend our knowledge on the full weight and height distribution for four rounds of AES.
References

Zero-Correlation Linear Cryptanalysis of Reduced-Round LBlock

Hadi Soleimany · Kaisa Nyberg

Abstract Zero-correlation linear attack is a new method for cryptanalysis of block ciphers developed by Bogdanov et al. in 2012. In this paper we adapt the matrix method to find zero-correlation linear approximations. Then we present several zero-correlation linear approximations for 14 rounds of LBlock and describe a cryptanalysis for 22 rounds of the reduced LBlock. After biclique attacks on LBlock revealed weaknesses in its key schedule, its designers presented a new version of the cipher with a revised key schedule. The attack presented in this paper is applicable to LBlock structure independently of the key scheduling. The attack needs distinct known plaintexts which is a more realistic attack model in comparison with impossible differential cryptanalysis which uses chosen plaintext pairs. Moreover, we performed simulations on an small variant LBlock and present the first experimental results on the theoretical model of the multidimensional zero-correlation linear cryptanalysis method.

Keywords block cipher, zero-correlation linear cryptanalysis, LBlock, matrix method

1 Introduction

Linear cryptanalysis is one of the most prominent cryptanalysis methods against block ciphers. Several extensions of linear cryptanalysis have been introduced which usually exploit several linear approximations with high correlation simultaneously. Kaliski and Robshaw used multiple linear approximations with the same key mask [7]. The concept of linear hull presented by Nyberg which
Table 1 Summary of the Attacks on LBlock

<table>
<thead>
<tr>
<th>Attack</th>
<th>Rounds</th>
<th>Data</th>
<th>Time</th>
<th>Memory (Bytes)</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral Attack (CP)</td>
<td>20</td>
<td>$2^{63}$</td>
<td>$2^{63}$.7</td>
<td>Not Specified</td>
<td>[16]</td>
</tr>
<tr>
<td>Impossible Differential (CP)</td>
<td>20</td>
<td>$2^{63}$</td>
<td>$2^{63}$.7</td>
<td>$2^{60}$</td>
<td>[16]</td>
</tr>
<tr>
<td>Impossible Differential (CP)</td>
<td>21</td>
<td>$2^{62.5}$</td>
<td>$2^{62.5}$.7</td>
<td>$2^{64}$</td>
<td>[11]</td>
</tr>
<tr>
<td>Impossible Differential† (CP)</td>
<td>21</td>
<td>$2^{63}$</td>
<td>$2^{63}$.5</td>
<td>$2^{68}$</td>
<td>[8]</td>
</tr>
<tr>
<td>Impossible Differential† (CP)</td>
<td>22</td>
<td>$2^{58}$</td>
<td>$2^{79}.28$</td>
<td>$2^{68}$</td>
<td>[8]</td>
</tr>
<tr>
<td>Impossible Differential† (CP)</td>
<td>22</td>
<td>$2^{64}$</td>
<td>$2^{70}.54$</td>
<td>$2^{64}$</td>
<td>This paper</td>
</tr>
<tr>
<td>Impossible Differential† (CP)</td>
<td>22</td>
<td>$2^{64}$</td>
<td>$2^{70}.54$</td>
<td>$2^{64}$</td>
<td>This paper</td>
</tr>
<tr>
<td>Impossible Differential† (CP)</td>
<td>22</td>
<td>$2^{60}$</td>
<td>$2^{79}$</td>
<td>$2^{64}$</td>
<td>This paper</td>
</tr>
<tr>
<td>Biclique (KP)†</td>
<td>Full</td>
<td>$2^{52}$</td>
<td>$2^{78}.4$</td>
<td>Negligible</td>
<td>[15]</td>
</tr>
</tbody>
</table>

† - this attack is applicable just on old version of the cipher, CP – Chosen Plaintexts, DKP – Distinct Known Plaintexts, KP – Known Plaintext

uses several linear approximation with the same input and output masks [13]. Multiple linear approximations cryptanalysis and multidimensional linear cryptanalysis are proposed in [2] and [6] respectively. Recently, a novel extension of linear cryptanalysis was proposed which uses zero-correlation linear approximations [4]. It can be seen as the counterpart of impossible differential cryptanalysis. The original proposal had the disadvantage to require almost the full codebook of data. Bogdanov et.al. proposed a framework which uses several independent zero-correlation linear approximations to reduce data complexity [5]. Based on the multidimensional linear attack, a new distinguisher was recently proposed to eliminate the independence assumption [3]. The distinguisher is supposed to use distinct known plaintexts.

In this paper the multidimensional zero-correlation linear method is applied to attack 22 rounds of LBlock [16]. LBlock is a lightweight block cipher with semi-Feistel structure. The security of the cipher has been evaluated in [8,11,16]. The designers proposed integral and impossible differential cryptanalysis up to 20 rounds of LBlock. Using low diffusion of key schedule, an improved impossible differential cryptanalysis has been applied up to 22 rounds of LBlock [8]. The attack uses $2^{58}$ chosen plaintexts and the time complexity is $2^{79}.28$ which is almost equivalent to the exhaustive search. Since LBlock had been designed before biclique cryptanalysis introduction, the designers re-evaluated the security of LBlock and showed that the cipher is vulnerable against biclique cryptanalysis. They proposed a modified key schedule algorithm to improve the security of LBlock [15] such that the previous attacks are not applicable. In this paper, we show how to use the matrix method [9,10] to find $8 \times 8$ different classes of zero-correlation linear approximations for 14 rounds which each one includes $2^8 - 1$ different zero-correlation approximations. Based on $2^8 - 1$ zero-correlation approximations we present an attack on 22 rounds of the reduced LBlock. This attack does not depend on the key schedule and it is applicable to both versions of LBlock. It exploits weaknesses in the permutation layer of LBlock to decrease the time complexity. The attack uses distinct known plaintexts. As depicted in Table 1, there is a trade-off between the time complexity and the data complexity of the attack.
The paper is structured as follows: In Section 2, we briefly describe LBlock. In Section 3 we review the previous work on zero-correlation linear cryptanalysis. In Section 4 we show how to use the matrix method as an automatic tool to find zero-correlation approximations and obtain several zero-correlation linear distinguisher for 14 rounds of the LBlock. Section 5 describes an attack on 22 rounds of LBlock. We conclude the paper in Section 6.

2 A Brief Description of LBlock

2.1 Notation

Throughout this paper we use the following notations:

− $Sk_i$ : 32-bit round key
− $\ll i$ : $i$-bit left cyclic shift
− $X(i)$ : $i$-th nibble of $X$ where the right most one is 0
− $X(i-j)$ : concatenation of $i, i-1, \cdots, j$-th nibble of $X$ where $i \geq j$
− $|\text{: concatenation of two binary strings}$
− $L_i|R_i$ : the output of the $i$-round of LBlock

2.2 LBlock Description

LBlock is a variant Feistel block cipher with 32 rounds. It supports 80 secret key bits and the block size is $b = 64$ bits. Each round includes 8 different $4 \times 4$ S-boxes and simple nibble-wise permutation. One round of LBlock and the round function are depicted in Figure 1.

**Encryption Algorithm** Let $P = L_0|R_0$ be a 64-bit plaintext. Then encryption process is as follows:

- For $i = 1, 2, \cdots, 31$, do
  - $R_i = L_{i-1}$
  - $L_i = F(L_{i-1}, SK_i) \oplus (R_{i-1} \ll 8)$
- $L_{32} = L_{31}$, $R_{32} = F(L_{31}, SK_{32}) \oplus (R_{31} \ll 8)$
- $C = L_{32}|R_{32}$.
In our attacks on reduced-round LBlock we also consider the last round to be without swapping the halves as in the original LBlock.

**Key schedule** The 80-bit master key $K$ is stored in a key register. In the $i$-th step, the leftmost 32 bits of current content of register $K$ are extracted as the round key $SK_i$. Then the key register is updated in each round. We do not exploit any property of the key scheduling. For the updating procedure in the original LBlock and the new version we refer to [16] and [15], respectively.

### 3 Zero-Correlation Linear Approximation

Consider a function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ and let the input of the function be $x \in \mathbb{F}_2^n$. A linear approximation with an input mask $u$ and an output mask $v$ is the following function:

$$ x \mapsto u \cdot x \oplus v \cdot f(x). $$

The linear approximation has probability

$$ p(u; v) = Pr(u \cdot x \oplus v \cdot f(x) = 0) $$

and its correlation is defined as follows:

$$ c_f(u; v) = 2p(u; v) - 1. $$

In linear cryptanalysis we are interested in the linear approximation with correlation far from zero. The number of known plaintexts needed in the linear cryptanalysis is inversely proportional to the squared correlation. Zero-correlation linear cryptanalysis uses linear approximations such that the correlation is equal to zero for all keys. If the number of zero-correlation approximations is $2^m$, then by [3] the number of required distinct plaintexts is about $2^{n+3-m/2}$. The key recovery can be done with the same method utilized by Matsui’s Algorithm 2 [12].

To describe this process in more detail, let us describe a cipher $E$ as a cascade $E = Ef \circ Ez \circ Eb$. Assume there exists $m$ independent linear approximations for $E_z$ such that all $\ell = 2^m - 1$ nonzero linear combinations of them have correlation zero. For each key candidate, the adversary encrypts the plaintexts for the beginning rounds $Eb$ and decrypts the corresponding ciphertexts for the final rounds $Ef$.

For each of $i \in \mathbb{F}_2^n$ he allocates a counter $T_i$ and computes the number of times which the corresponding data value is equal to $i$. Then the adversary computes the statistic $T$ value

$$ T = \sum_{i=0}^{2^m-1} \frac{(T_i - N2^{-m})^2}{N2^{-m}(1 - 2^{-m})}. \quad (1) $$

The value $T$ for right key guess follows a $\chi^2$-distribution with mean $\mu_0 = \ell \frac{2^m - N}{2^m - 1}$ and variance $\sigma_0^2 = 2\ell \left( \frac{2^m - N}{2^m - 1} \right)^2$ while for the wrong key the distribution is a $\chi^2$-distribution with mean $\mu_1 = \ell$ and variance $\sigma_1^2 = 2\ell$. 

4
Let show error probability type I as $\alpha$ and error probability type II as $\beta$. If we consider the decision threshold $t = \mu_0 + \sigma_0 z_{1-\alpha} = \mu_1 + \sigma_1 z_{1-\beta}$ then the amount of distinct known plaintexts is as follows:

$$N = 2^n \frac{(z_{1-\alpha} + z_{1-\beta})}{\sqrt{\ell/2 - z_{1-\beta}}}$$  (2)

where $z_p = \Phi^{-1}(p)$ for $0 < p < 1$ where $\Phi$ is the cumulative function of the standard normal distribution. For more details we refer to [3].

4 The Matrix Method

Several tools have been proposed for finding statistical distinguisher. Such tools help us to analyze algorithms systematically. A cryptanalytic tool for finding impossible differential characteristics in block ciphers with bijective function was introduced in [9,10]. It is called the matrix method and uses the miss-in-the-middle approach to find impossible differential characteristic. The miss-in-the-middle technique proposes to construct the impossible differential characteristic by two (truncated) differential paths with probability one and which lead to a contradiction in the middle. The matrix method is a tool for finding these paths. In this section we show this technique is also useful for finding zero-correlation linear approximation. We can follow the linear approximation patterns of input and output masks in the intermediate rounds and inquire whether no linear characteristics with non-zero-correlation exists. So the matrix method is also useful to automate the process of finding the longest zero-correlation linear approximations.

4.1 Matrix Method for Finding Linear Approximation with Correlation Zero

The state is partitioned into $n$ words (usually of the same length). In the linear approximation, the linear masks applied to the words can be of the following five types:

1. zero mask, denoted by 0,
2. an arbitrary non-zero mask, denoted by $\overline{a}$,
3. non-zero mask with a fixed value $a$, denoted by $a$,
4. the exclusive-or of a fixed non-zero mask $a$ and an arbitrary non-zero mask, denoted by $\overline{a}$,
5. any mask, denoted by $\ast$.

After that we describe the encryption round as a matrix $M^{n \times n}$. The matrix shows how a linear mask of each output word is affected by the linear mask of an input word. Let show the input and the output of the round by two bit strings $A$ and $B$ respectively. If $B(j)$ is not affected by a linear mask of $A(i)$ the value $(i, j)$ set to 0. If a linear mask of $A(i)$ affects $B(j)$ directly the value $(i, j)$ set to 1. Finally if $B(j)$ is affected by a linear mask of $A(i)$ after the round
Table 2 Arithmetic rules. The table of the left gives the addition rules between two mask types. The table on the right shows the operation rules of multiplication by 0, 1 and \( 1_F \).

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>0</th>
<th>a</th>
<th>( \bar{a} )</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>a</td>
<td>( \bar{a} )</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>a</td>
<td>( \bar{a} )</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>a + b</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1_F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>( \bar{a} )</td>
<td>0</td>
<td>( \bar{a} )</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

function the value \((i,j)\) set to \(1_F\). For decryption of the round, another matrix is defined similarly. To define the matrices we can use lemmas in Appendix A which are introduced in [4] (see also [1]).

We can apply the exclusive-or operation to the five types of masks defined above resulting in arithmetic rules. These rules are given in the table on the left of Table 2. On the right hand side of Table 2 we show how certain operations will modify the five types of masks. The operations are: multiplication by 0, multiplication by 1 and multiplication by \(1_F\). The last operation means that, given a bijection \(F\) and a type \(\tau\) of masks, \(\tau \cdot 1_F\) is the type of masks \(v\) for which there is \(u\) of type \(\tau\) such that the linear approximation \(u \cdot x + v \cdot F(x) = 0\) holds with non-zero correlation.

For a given state, we use the matrix iteratively to obtain the new state over multiple rounds. To find the longest zero-correlation linear approximation we compute the new states in both forward and backward directions just before the values of all words become only \(*\). Finally, we scan intermediate values and check the incoherence of events.

For example the encryption matrix of Feistel structure is \(\begin{pmatrix} 0 & 1 \\ 1 & 1_F \end{pmatrix}\). If we assume the initial mask type as \((a, 0)\) the mask type of the third round can be obtained as follows:

\[
(a, 0) \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1_F \end{pmatrix} = (a \cdot 0 + 0 \cdot 1, a \cdot 1 + 0 \cdot 1_F) = (0 + 0, a + 0) = (0, a)
\]

\[
(0, a) \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1_F \end{pmatrix} = (0 \cdot 0 + a \cdot 1, 0 \cdot 1 + a \cdot 1_F) = (0 + a, 0 + \bar{a}) = (a, \bar{a})
\]

\[
(a, 1) \cdot \begin{pmatrix} 0 & 1 \\ 1 & 1_F \end{pmatrix} = (a \cdot 0 + \bar{a} \cdot 1, a \cdot 1 + \bar{a} \cdot 1_F) = (0 + \bar{a}, a + \bar{a}) = (\bar{a}, \pi)
\]

4.2 Zero-Correlation Linear Approximation for 14-rounds of LBlock

We applied the matrix method for LBlock. The round matrices for encryption and decryption can be found in Appendix B. The longest zero-correlation linear
approximation was obtained for 14 rounds of LBlock. If the input mask would be exactly one non-zero nibble in \( L_r \) and the output mask after 14 rounds would be one non-zero nibble in \( R_{r+14} \), then the linear approximation has correlation zero. For example \((00000000\mid00000000)\rightarrow(00000000\mid00000000)\) has correlation exactly zero which the values \( a \) and \( b \) are non zero. The development of the states of mask types as encryption proceeds from round to round is depicted in Table 3. The contradiction occurs in \( R_7(5) \). We note there exists \( 8 \times 8 \) different classes of zero-correlation linear approximations for 14 rounds each of which includes \( 2^8 - 1 \) different zero-correlation approximations. We will use this observation to build a multidimensional linear approximation to minimize the data complexity as described in Section 3.

<table>
<thead>
<tr>
<th>Round</th>
<th>( T_{L_r} )</th>
<th>( T_{R_s} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>1</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>2</td>
<td>0a000000</td>
<td>00000000</td>
</tr>
<tr>
<td>3</td>
<td>00000000</td>
<td>0a000000</td>
</tr>
<tr>
<td>4</td>
<td>0000000a</td>
<td>00000000</td>
</tr>
<tr>
<td>5</td>
<td>00000000</td>
<td>0000000a</td>
</tr>
<tr>
<td>6</td>
<td>00000000</td>
<td>00000000+</td>
</tr>
<tr>
<td>7</td>
<td>00000000</td>
<td>0+0a0000+</td>
</tr>
<tr>
<td>8</td>
<td>00000000+</td>
<td>00000000+</td>
</tr>
<tr>
<td>9</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>10</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>11</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>12</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>13</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td>14</td>
<td>00000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

5 Zero-Correlation Linear Cryptanalysis of 22 Reduced-Round LBlock

In this section, we propose a zero-correlation linear attack on 22-round LBlock. The attack utilizes the 14-round zero-correlation linear approximations described in Table 3 from round 5 to 18. After collecting sufficient plaintext-ciphertext pairs, we guess corresponding subkeys for the first four rounds and the last four rounds and estimate the correlation of approximations as described in Algorithm 1 in Appendix C.

Based on the error probabilities \( \alpha \) and \( \beta \), the number of pairs \( N \) in Algorithm 1 and the decision threshold \( t \) are determined. The time complexity of the Algorithm 1 is \( N \cdot 2^{28} \cdot 2^{28} \) where \( N \) is the number of plaintexts used in the cryptanalysis. So the time complexity is much more than exhaustive search. To overcome this restriction we note \( L_4(4) \) and \( R_{18}(6) \) are not affected by all
We call these bits active bits in rounds 1 – 4 and 19 – 22. So Algorithm 1 is not optimal and it repeats the same procedure for different pairs. We show that it is possible to remove repetitions and reduce time complexity significantly.

The nibble $L_4(4)$ is affected by 32 bits of plaintext $L_0|R_0$, 20 bits of $L_1|R_1$, 12 bits of $L_2|R_2$ and 8 bits of $L_3|R_3$. Also $L_{18}(6)$ is affected by 32 bits of ciphertext $L_{22}|R_{22}$, 20 bits of $L_{21}|R_{21}$, 12 bits of $L_{20}|R_{20}$ and 8 bits of $L_{19}|R_{19}$. We call these bits active and other ones neutral. The idea is to ignore neutral bits and instead of encrypting and decrypting all plaintext-ciphertext pairs, do it only once and count the number of pairs, which have the same value in active bits. In each step, for each subkey candidate, we encrypt (decrypt) active bits in round $r$ over one round and count the number of pairs which give the same value in active bits in round $r + 1$ ($r - 1$).

**Attack procedure**

The attack procedure is as follows:

1. Collect $N$ plaintexts with corresponding ciphertexts.
2. Allocate a 8-bit counter $N_0[x_0, x_{22}]$ for each of $2^{64}$ possible values of $(x_0|x_{22})$ where $x_0 = L_0(5, 4, 2, 1, 0)|R_0(6, 4, 1)$ and $x_{22} = L_{22}(7, 6, 4, 2, 0)|R_{22}(7, 5, 2)$ and set them zero. Calculate the number of pairs of plaintext-ciphertext with given values $x_0$ and $x_{22}$ and save it in $N_0[x_0, x_{22}]$. In this step, around $2^{64}$ plaintext-ciphertext pairs are divided into $2^{64}$ different state. The expected pairs for each state is around one. So the assumption $N_0$ as a 8-bit counter is sufficient.
3. Guess the 3 nibbles $SK_1(4, 2, 1)$. Allocate a counter $N_1[x_1, x_{22}]$ for each of $2^{52}$ possible values of $(x_1|x_{22})$ where $x_1 = L_1(6, 3, 0)|R_1(5, 0)$ and set them zero. For all $2^{32}$ possible values of $x_0$, encrypt $x_0$ one round to obtain $x_1$ and update the value $N_1[x_1, x_{22}] = N_1[x_1, x_{22}] + N_0[x_0, x_{22}]$ for all $2^{32}$ values of $x_{22}$.
4. Guess 2 nibbles $SK_2(6, 0)$. Allocate a counter $N_2[x_2, x_{22}]$ for each of $2^{44}$ possible values of $(x_2|x_{22})$ where $x_2 = L_2(7, 2)|R_2(3)$ and set them zero. For all $2^{20}$ possible values of $x_1$, encrypt $x_1$ one round to obtain $x_2$ and update the value $N_2[x_2, x_{22}] = N_2[x_2, x_{22}] + N_1[x_1, x_{22}]$ for all $2^{20}$ values of $x_{22}$.
5. Guess the nibble $SK_3(7)$. Allocate a counter $N_3[x_3, x_{22}]$ for each of $2^{40}$ possible values of $(x_3|x_{22})$ where $x_3 = L_3(5)|R_3(2)$ and set them zero. For all $2^{12}$ possible values of $x_2$, encrypt $x_2$ one round to obtain $x_3$ and update the value $N_3[x_3, x_{22}] = N_3[x_3, x_{22}] + N_2[x_2, x_{22}]$ for all $2^{12}$ values of $x_{22}$.
6. Guess the nibble $SK_4(5)$. Allocate a counter $N_4[x_4, x_{22}]$ for each of $2^{36}$ possible values of $(x_4|x_{22})$ where $x_4 = L_4(4)$ and set them zero. For all $2^8$ possible values of $x_3$, encrypt $x_3$ one round to obtain $x_4$ and update the value $N_4[x_4, x_{22}] = N_4[x_4, x_{22}] + N_3[x_3, x_{22}]$ for all $2^8$ values of $x_{22}$.
7. Guess the 3 nibbles $SK_{22}(7, 6, 0)$. Allocate a counter $N_5[x_4, x_{21}]$ for each of $2^{24}$ possible values of $(x_4|x_{21})$ where $x_{21} = L_{21}(4, 2)|R_{21}(5, 3, 0)$ and set them zero. For all $2^{12}$ possible values of $x_{22}$, decrypt $x_{22}$ one round to obtain $x_{21}$ and update the value $N_5[x_4, x_{21}] = N_5[x_4, x_{21}] + N_4[x_4, x_{22}]$ for all $2^4$ values of $x_4$. 

8
8. Guess 2 nibbles $SK_{21}(5, 0)$. Allocate a counter $N_6[x_4, x_{20}]$ for each of $2^{16}$ possible values of $(x_4, x_{20})$ where $x_{20} = L_{20}(3)|R_{20}(2, 0)$ and set them zero. For all $2^{20}$ possible values of $x_{21}$, decrypt $x_{21}$ one round to obtain $x_{20}$ and update the value $N_6[x_4, x_{20}] = N_6[x_4, x_{20}] + N_5[x_4, x_{21}]$ for all $2^4$ values of $x_4$.

9. Guess the nibble $SK_{20}(2)$. Allocate a counter $N_7[x_4, x_{19}]$ for each of $2^{12}$ possible values of $(x_4|x_{19})$ where $x_{19} = L_{19}(0)|R_{19}(1)$ and set them zero. For all $2^{12}$ possible values of $x_{20}$, decrypt $x_{20}$ one round to obtain $x_{19}$ and update the value $N_7[x_4, x_{19}] = N_7[x_4, x_{19}] + N_6[x_4, x_{20}]$ for all $2^4$ values of $x_4$.

10. Guess the nibble $SK_{19}(1)$. Allocate a counter $N_8[x_4, x_{18}]$ for each of $2^8$ possible values of $(x_4|x_{18})$ where $x_{18} = R_{18}(6)$ and set them zero. For all $2^8$ possible values of $x_{19}$, decrypt $x_{19}$ one round to obtain $x_{18}$ and update the value $N_8[x_4, x_{18}] = N_8[x_4, x_{18}] + N_7[x_4, x_{19}]$ for all $2^4$ values of $x_4$.

11. Compute the statistic value $T = N \cdot 2^8 \sum_{x_4=1}^{2^8} \sum_{x_{18}=0}^{2^8-1} \left( \frac{N_8[x_4, x_{18}]}{N} - \frac{1}{2^6} \right)$. If $T < t$, then the guess key is a possible candidate.

12. Do exhaustive search for all keys which corresponds to the guess subkey bits.

**Attack complexity**

The memory complexity of the attack is dominated by step 2 which needs $2^{64}$ bytes. Time complexity of step 1 and 2 is equal to the number of needed plaintext-ciphertext pairs $N$. The time complexity in each step between 3 and 11 depends on the number of accesses to the memory. The time complexity for each round is listed separately in Appendix D. Also step 12 requires $2^{60} \cdot \beta$ full encryption because we expect a wrong subkey survives with probability $\beta$.

The time complexity is dominated by step 3 and step 12. The time complexity of round 3 is $2^{70}$ memory accesses. If we consider one memory accesses as a half round, the time complexity of step 3 is $2^{70} \times \frac{1}{2} \times \frac{1}{22} = 2^{70.54}$ of 22-round LBlock. Based on the error probability type I $\alpha$ and error probability type II $\beta$, the number of plaintexts-ciphertext pairs needed, time complexity of step 12 and success probability are determined.

There is a trade-off between the time complexity and the data complexity of the attack, as depicted in Table 1. To reduce the time complexity as much as possible, we assume to have access the full codebook. In this case, the error probabilities and time complexity of step 12 is negligible compared to the complexity of step 3. To have a lowest data complexity, we can set $\alpha = 2^{-2.7}$ and $\beta = 2^{-1}$. In this case data complexity decreases to $N = 2^{60}$ in cost of increasing time complexity. The time complexity is dominated by step 12 which needs $2^{60} \cdot 2^{-1} = 2^{70}$ 22-round LBlock encryption. The more realistic assumption is the state between these cases. For example, if we set $\alpha = 2^{-2.7}$ and $\beta = 2^{-10}$ then $z_{1-\alpha} = 1$ and $z_{1-\beta} = 3.09$. Equation (2) determines the data complexity $N = 2^{62.1}$. The time complexity is dominated by step 3 and 12 $2^{70.5} + 2^{70} = 2^{71.27}$. The success probability is $1 - \alpha = 0.84$. 

9
6 Experimental Results

As noted in [16] LBlock can be described as a 16-branch generalized Feistel cipher (GFC) with improved permutation. The nibble-wise permutation has been chosen such that it achieves the best diffusion as proposed in [14]. To verify the theoretical model of zero-correlation attacks [3] we implement the described attack on an small variant of LBlock with block length 32-bit. Two optimal word-wise permutations for improved Type II GFC with 8 branches were suggested in [14]. To make the small variant cipher similar as much as possible to the original one, we choose the permutation which is not based on the Bruijn graph. We consider the key schedule like original LBlock and choose the leftmost 16 bits of the register $K$ as the round key. Matrix method produces a similar multidimensional linear approximation with correlation zero for 10 rounds of the small variant of LBlock cipher. This linear approximation has exactly one non-zero nibble in the input mask and one non-zero nibble in the output mask.

To evaluate a 10-round distinguisher from round 2 to 12, we consider 14 rounds of the small LBlock. The distinguisher depends on 2-nibbles of subkeys in the first two rounds and 2-nibbles of subkeys in the last two rounds. We consider 30 different sets of distinct known plaintexts with different secret keys. In each experiment the behavior of the statistic $T$ in Equation 1 is studied for the right key and also for (just) one wrong key. The results of the implementation is shown in Figure 2. As predicted by the theoretical model, when more than $2^{30.2}$ distinct known plaintexts are used, the correct key is very likely to pass the test, while the wrong keys would fail. Access to the full codebook leads to the key recovery with negligible error probability. When us-
ing \(2^{28}\) distinct known plaintexts, the right key survives with high probability but several wrong keys remain too.

### 7 Conclusion

In this paper we showed how to use the matrix method to establish zero-correlation linear approximations automatically. We used this method to obtain several zero-correlation linear approximations over 14 rounds of LBlock. We believe that the described method will be useful for analysis of other block ciphers, too. Based on the 14-round distinguisher we present an attack on 22 rounds of LBlock. While the previous attack, which can break the same number of rounds, uses chosen plaintext pairs, our attack assumes only that the plaintexts are distinct. Unlike biclique techniques, the proposed cryptanalysis does not exploit the structure of the key schedule and therefore applies also to the new version of LBlock. Finally, we implement the attack for an small variant of LBlock and run simulations to experimentally validate the statistical model of zero-correlation linear cryptanalysis presented in [3].

**Acknowledgements**  We wish to thank the anonymous reviewers for useful comments. The work of Hadi Soleimany is supported by Helsinki Doctoral Program in Computer Science - Advanced Computing and Intelligent Systems (HECSE) and partly supported by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

**References**

A Lemmas

To describe the encryption or decryption round, we can use the following lemmas:

**Lemma 1 XOR operation:** Let \( f(x_1, x_2) = x_1 \oplus x_2 \) then the correlation of linear approximation \( u_1 \cdot x_1 + u_2 \cdot x_2 = v \cdot f(x_1, x_2) \) is non-zero if and only if \( u_1 = u_2 = v \).

**Lemma 2 Branching operation:** Let \( f(x) = (x, x) \) then the correlation of linear approximation \( u_1 \cdot x + u_2 \cdot x = v \cdot f(x) \) is non-zero if and only if \( u = u_1 + u_2 \).

**Lemma 3 Bijective function:** Let \( f(x) \) be a bijective function then the correlation of linear approximation \( u \cdot x = v \cdot f(x) \) is non-zero if and only if \( u = v = 0 \) or \( u \neq 0 \) and \( v \neq 0 \).

B LBlock Matrices

\[
M_{Encryption} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1_F
\end{pmatrix}
\]
C Basic Key Recovery Attack on 22 Reduced-Round LBlock

Algorithm 1 Attack procedure

for all $2^{28}$ subkey nibbles $SK_1(4, 2, 1), SK_2(6, 0), SK_3(7), SK_4(5)$ in rounds 1−4 do
for all $2^{32}$ subkey nibbles $SK_{19}(1), SK_{20}(2), SK_{21}(5, 0), SK_{22}(7, 6, 0)$ in rounds 19−22 do
for all $2^8$ possible values $i = 0, \ldots, 2^8 - 1$ do
allocate the counter $T_i$ and set them zero
end for
for all $N$ plaintext-ciphertext pairs do
encrypt plaintext to obtain the nibble $L_{4}(4)$;
decrypt ciphertext to obtain the nibble $R_{18}(6)$;
for corresponding $i = (L_{4}(4)|R_{18}(6))$ increase the counter $T_i$ by one.
end for
compute the statistic value $T = N \cdot 2^8 \sum_{i=0}^{2^8-1} \left( T_i - \frac{1}{2^8} \right)$
If $T < t$, then the guess key is a possible candidate.
end for
end for
Do exhaustive search for all keys which corresponds to the guess subkey bits

D Attack Complexity

The time complexity of steps 3-11 in the described attack in Section 5 is as follows:

Step 3 requires $2^{12} \times 2^{12} \times 2^{32} = 2^{76}$ memory accesses, because we should guess 12 bits for $SK_1$, and for $2^{32}$ values encrypt $x_0$ one round and then update $N_1$ for $2^{32}$ times.

Step 4 requires $2^{12} \times 2^6 \times 2^{20} \times 2^{12} = 2^{72}$ memory accesses, because for all of guessed $2^{12}$ keys in previous step, we should guess 8 bits for $SK_2$, and for $2^{20}$ values encrypt $x_1$ one round and then update $N_2$ for $2^{32}$ times.

Step 5 requires $2^{20} \times 2^4 \times 2^{12} \times 2^{12} = 2^{68}$ memory accesses, because for all of guessed $2^{20}$ keys in previous steps, we should guess 4 bits for $SK_3$ and for $2^{12}$ values encrypt $x_2$ one round and then update $N_3$ for $2^{32}$ times.

Step 6 requires $2^{24} \times 2^4 \times 2^8 \times 2^{32} = 2^{68}$ memory accesses, because for all of guessed $2^{24}$ keys in previous steps, we should guess 4 bits for $SK_4$ and for $2^8$ values encrypt $x_3$ one round and then update $N_4$ for $2^{32}$ times.
Step 7 requires $2^{28} \times 2^{12} \times 2^{1} = 2^{72}$ memory accesses, because for all of guessed $2^{28}$ keys in previous steps, we should guess 12 bits for $SK_{22}$ and for $2^{32}$ values decrypt $x_{22}$ one round and then update $N_{5}$ for $2^{4}$ times.

Step 8 requires $2^{40} \times 2^{8} \times 2^{1} = 2^{72}$ memory accesses, because for all of guessed $2^{40}$ keys in previous steps, we should guess 8 bits for $SK_{21}$ and for $2^{20}$ values decrypt $x_{21}$ one round and then update $N_{6}$ for $2^{4}$ times.

Step 9 requires $2^{48} \times 2^{4} \times 2^{8} = 2^{68}$ memory accesses, because for all of guessed $2^{48}$ keys in previous steps, we should guess 4 bits for $SK_{20}$ and for $2^{12}$ values decrypt $x_{20}$ one round and then update $N_{7}$ for $2^{4}$ times.

Step 10 requires $2^{52} \times 2^{4} \times 2^{8} \times 2^{4} = 2^{68}$ memory accesses, because for all of guessed $2^{52}$ keys in previous steps, we should guess 4 for $SK_{19}$ and for $2^{8}$ values decrypt $x_{19}$ one round and then update $N_{8}$ for $2^{4}$ times.

Step 11 requires $2^{56} \times 2^{8} = 2^{64}$ memory accesses, because for all of guessed $2^{56}$ keys in previous steps, we should read $2^{8}$ values of $N_{8}[x_{4}, x_{18}]$. 
A Polynomial Time Version of LLL With Deep Insertions

Felix Fontein · Michael Schneider · Urs Wagner

February 16, 2013

Abstract Lattice reduction algorithms have numerous applications in number theory, algebra, as well as in cryptanalysis. The most famous algorithm for lattice reduction is the LLL algorithm. In polynomial time it computes a reduced basis with provable output quality. One early improvement of the LLL algorithm was LLL with deep insertions (DeepLLL). The output of this version of LLL has higher quality in practice but the running time seems to explode. Weaker variants of DeepLLL, where the insertions are restricted to blocks, behave nicely in practice concerning the running time. However no proof of polynomial running time is known. In this paper a new variant of DeepLLL with provably polynomial running time is presented. We compare the practical behavior of the new algorithm to classical LLL, BKZ as well as blockwise variants of DeepLLL regarding both the output quality and running time.

Keywords Lattice Reduction · LLL Algorithm · Deep Insertion

Mathematics Subject Classification (2000) 68R05 · 94A60 · 68R05 · 94A60

1 Introduction

The well-known LLL lattice reduction algorithm was presented in 1982 by Lenstra, Lenstra, Lovász [LLL82]. Apart from various other applications (e.g. [NV10, Chapter 9,10]) it has already at an early stage been used to attack various public key cryptosystems. Nevertheless lattice problems remain popular when it comes to the construction of provably secure cryptosystems (e.g. [NY10, Chapter 13]). Consequently improvements in lattice reduction still have a direct impact on the security of many cryptosystems and rise high interest in the crypto-community.

Many lattice reduction algorithms used in practice are generalizations of the LLL algorithm. The Block-Korkine-Zolotarev (BKZ) reduction algorithm by Schnorr and Euchner [SE94] is probably
the most used algorithm when stronger reduction than the one achieved by LLL is required. It can be seen as a generalization of LLL to higher block sizes, and while the running time seems to behave well for small block sizes [GN08], no useful upper bound has been proven so far. Another improvement of the LLL algorithm has also been suggested in [SE94]. While in LLL adjacent basis vectors are swapped if certain conditions are satisfied, in the so called LLL with deep insertions (DeepLLL in the sequel), basis vectors can be swapped even when not adjacent. The practical behavior of DeepLLL when it comes to the reducedness of the output basis is superior to the one of LLL. Unfortunately also the running time explodes and does not seem to be polynomial in the dimension of the lattice. One attempt to get around this problem is to restrict the insertions to certain blocks of basis vectors. While the authors in [SE94] claim that these blockwise restriction variants of DeepLLL run in polynomial time, we refer to [NS06,GN08]. There the practical behavior of the reduction algorithms is investigated using the widely used NTL library.

In this paper we present a new version of DeepLLL, called PotLLL. To our knowledge it is the first improvement of LLL with regard to deep insertions which provably runs in polynomial time. The practical behavior of PotLLL regarding both the output quality and running time is empirically tested and compared to BKZ and DeepLLL with different block sizes. The tests are performed with a completely new implementation of the different reduction algorithms. This additionally allows an independent review of the results in [NS06,GN08]. The tests indicate that our algorithm can serve as a serious alternative to BKZ with low block sizes.

The paper is organized as follows. In Section 2 all necessary notations and definitions are given. In Section 3 the reduction notion and the new algorithm is presented and a theoretical analysis is provided. Section 4 contains the empirical results and conclusions are drawn in Section 5.

2 Preliminaries

A lattice \( \mathcal{L} \subset \mathbb{R}^m \) of rank \( n \) and dimension \( m \) is a discrete subgroup of \( \mathbb{R}^m \) generated by integer linear combinations of \( n \) linearly independent vectors \( b_1, \ldots, b_n \) in \( \mathbb{R}^m \):

\[
\mathcal{L} = \langle b_1, \ldots, b_n \rangle := \left\{ \sum_{i=1}^n x_i b_i \mid \forall i : x_i \in \mathbb{Z} \right\}.
\]

We will often write the basis \( b_1, \ldots, b_n \) as rows of a matrix \( B \) in the following way \( B = [b_1, \ldots, b_n] \).

In order to have exact representations in computers, only lattices in \( \mathbb{Q}^n \) are considered. Simple scaling by the least common multiple of the denominators allows us to restrict ourselves to integer lattices \( \mathcal{L} \subseteq \mathbb{Z}^m \). The volume of a lattice \( \mathcal{L}(B) \) equals the volume of its fundamental parallelepiped \( \text{vol}(\mathcal{L}) = \sqrt{\det(B^*B)} \). For \( n \geq 2 \), a lattice has infinitely many bases as \( \mathcal{L}(B) = \mathcal{L}(B') \) if and only if \( \exists U \in GL_n(\mathbb{Z}) : B = U B' \). Therefore, the volume of a lattice is well defined. By \( \pi_k : \mathbb{R}^m \rightarrow \text{span}\{b_1, \ldots, b_{k-1}\} \) denotes the orthogonal projection from \( \mathbb{R}^m \) onto the orthogonal complement of span \( \{b_1, \ldots, b_{k-1}\} \). In particular, \( \pi_1 = \text{id}_{\mathbb{R}^m} \) and \( b_i^* := \pi_i(b_i) \) equals the \( i \)-th basis vector of the Gram-Schmidt orthogonalization \( B^* = [b_1^*, \ldots, b_n^*] \) of \( B \). By \( \mu_{i,j} := \langle b_i, b_j^* \rangle / \langle b_j^*, b_j^* \rangle \), \( j < i \), we denote the Gram-Schmidt coefficients. The Gram-Schmidt vectors can iteratively be computed by \( \pi_i(b_k) = b_k^* = b_k - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \).

Throughout this paper, by \( \| \cdot \| \) we denote the Euclidean norm and by \( \lambda_1(\mathcal{L}) \) we denote the length of a shortest non-zero vector in \( \mathcal{L} \) with respect to the Euclidean norm: \( \lambda_1(\mathcal{L}) := \min_{v \in \mathcal{L} \setminus \{0\}} \|v\| \). Determining \( \lambda_1(\mathcal{L}) \) is commonly known as the shortest vector problem (SVP) and is proven to be NP-hard (under randomized reductions) (see e.g. [MG02]). Upper bounds with respect to the determinant exist, for all rank \( n \) lattices \( \mathcal{L} \) we have

\[
\frac{\lambda_1(\mathcal{L})^2 \text{vol}(\mathcal{L})^{2/n}}{\gamma_n} \leq 1 + \frac{n}{4},
\]

where \( \gamma_n \) is given by [NV10].
where \( \gamma_n \) is the Hermite constant in dimension \( n \). Given a relatively short vector \( v \in \mathcal{L} \), one measures its quality by the Hermite factor \( \|v\|/\text{vol}(\mathcal{L})^{1/n} \) it achieves. Modern lattice reduction algorithms achieve a Hermite factor which is exponential in \( n \) and no polynomial time algorithm is known to achieve linear or polynomial Hermite factors.

Let \( S_n \) denote the group of permutations of \( n \) elements. By applying \( \sigma \in S_n \) to a basis \( B = [b_1, \ldots, b_n] \), the basis vectors are reordered \( \sigma B = [b_{\sigma(1)}, \ldots, b_{\sigma(n)}] \). For \( 1 \leq k \leq \ell \leq n \) we define a class of elements \( \sigma_{k, \ell} \in S_n \) as follows:

\[
\sigma_{k, \ell}(i) = \begin{cases} 
  i & \text{for } i < k \text{ or } i > \ell, \\
  \ell & \text{for } i = k, \\
  i - 1 & \text{for } k < i \leq \ell. 
\end{cases}
\]

(2.1)

Note that \( \sigma_{k, \ell} = \sigma_{k, k+1} \sigma_{k+1, k+2} \cdots \sigma_{\ell-1, \ell} \) and that \( \sigma_{k, k+1} \) is swapping the two elements \( k \) and \( k + 1 \).

**Definition 2** Let \( \delta \in (1/4,1] \). A basis \( B = [b_1, \ldots, b_n] \) of a lattice \( \mathcal{L}(b_1, \ldots, b_n) \) is called \( \delta - \text{LLL reduced} \) if and only if it satisfies the following two conditions:

1. \( \forall 1 \leq j < i \leq n : |\mu_{i,j}| \leq \frac{1}{2} \) (size-reduced).
2. \( 1 \leq k < n : \delta \cdot \|\pi_k(b_k)\|^2 \leq \|\pi_k(b_{k+1})\|^2 \) (Lovász-condition).

A \( \delta - \text{LLL} \) reduced basis \( B = [b_1, \ldots, b_n] \) can be computed in polynomial time [LLL82] and provably satisfies the following bounds:

\[
\|b_1\| \leq (\delta - 1/4)^{-(n-1)/2} \cdot \lambda_1(\mathcal{L}(B)) \quad \text{and} \quad \|b_i\| \leq (\delta - 1/4)^{-(n-1)/4} \cdot \text{vol}(\mathcal{L}(B))^{1/n}. 
\]

(2.2)

While these bounds can be reached, they are worst case bounds. In practice, LLL reduction algorithms behave much better [NS06]. One early attempt to improve the LLL reduction algorithm is due to Schnorr and Euchner [SE94] who came up with the notion of a DeepLLL reduced basis:

**Definition 3** Let \( \delta \in (1/4,1] \). A basis \( B = [b_1, \ldots, b_n] \) of a lattice \( \mathcal{L}(b_1, \ldots, b_n) \) is called \( \delta - \text{DeepLLL reduced with blocksize } \beta \) if and only if it satisfies the following two conditions:

1. \( \forall 1 \leq j < i \leq n : |\mu_{i,j}| \leq \frac{1}{2} \) (size-reduced).
2. \( \forall 1 \leq k < \ell \leq n \) with \( k \leq \beta \vee \ell - k \leq \beta : \delta \cdot \|\pi_k(b_k)\|^2 \leq \|\pi_k(b_{\ell})\|^2 \).

If \( \beta = n \) we simply call this a DeepLLL reduced basis. While the first basis vector of DeepLLL reduced bases in the worst case does not achieve a better Hermite factor than classical LLL (see Section 3.4), the according reduction algorithms usually return much shorter vectors than pure LLL. Unfortunately no polynomial time algorithm to compute DeepLLL reduced bases is known.

The following definition is used in the proof (see e.g. [MC02]) of the polynomial running time of the LLL reduction algorithm and will play a main role in our improved variant of LLL.

**Definition 3** The potential \( \text{Pot}(B) \) of a lattice basis \( B = [b_1, \ldots, b_n] \) is defined as

\[
\text{Pot}(B) := \prod_{i=1}^{n} \text{vol}(\mathcal{L}(b_1, \ldots, b_i)) = \prod_{i=1}^{n} \|b_i\|^{2(n-i+1)}.
\]

Here it is used that \( \text{vol}(\mathcal{L}) = \prod_{i=1}^{n} \|b_i^*\| \). Note that, unlike the volume of the lattice, the potential of a basis is variant under basis permutations. The following lemma describes how the potential changes if \( \sigma_{k, \ell} \) is applied to the basis.

**Lemma 1** Let \( B = [b_1, \ldots, b_n] \) be a lattice basis. Then for \( 1 \leq k \leq \ell \leq n \)

\[
\text{Pot}(\sigma_{k, \ell}B) = \text{Pot}(B) \cdot \prod_{i=k}^{\ell} \frac{\|\pi_i(b_i)\|^2}{\|\pi_i(b_i)\|^2}.
\]
Proof First note that it is well-known that \( \text{Pot}(\sigma_{k,k+1}B) = \|\pi_k(b_{k+1})\|^2/\|\pi_k(b_k)\|^2 \cdot \text{Pot}(B) \). This property is used in the proofs of the polynomial running time of LLL [LLL82, MG02].

We prove the claim by induction over \( \ell - k \). The claim is true for \( k = \ell \). For \( k < \ell \), \( \sigma_{k,\ell} = \sigma_{k,k+1}\sigma_{k+1,\ell} \). As \( b_\ell \) is the \( (k+1) \)-th basis vector of \( \sigma_{k+1,\ell} \), with the above identity we get \( \text{Pot}(\sigma_{k,\ell}B) = \text{Pot}(\sigma_{k,k+1}\sigma_{k+1,\ell}B) = \|\pi_k(b_\ell)\|^2/\|\pi_k(b_k)\|^2 \cdot \text{Pot}(\sigma_{k+1,\ell}B) \), which completes the proof. \( \square \)

3 The Potential-LLL Reduction

In this section we present our polynomial time variant of DeepLLL. We start with the definition of a \( \delta \)-PotLLL reduced basis. Then we present an algorithm that outputs such a basis followed by a runtime proof.

Definition 4 Let \( \delta \in (1/4,1] \). A lattice basis \( B = [b_1,\ldots,b_n] \) is \( \delta \)-PotLLL reduced if and only if

1. \( \forall 1 \leq j < i \leq n : |\mu_{i,j}| \leq \frac{1}{\delta} \) (size-reduced).
2. \( \forall 1 \leq k < \ell \leq n : \delta \cdot \text{Pot}(B) \leq \text{Pot}(\sigma_{k,\ell}B) \).

Lemma 2 A \( \delta \)-PotLLL reduced basis \( B \) is also \( \delta \)-LLL reduced.

Proof Lemma [1] shows that \( \delta \cdot \text{Pot}(B) \leq \text{Pot}(\sigma_{k,k+1}B) \) if and only if \( \delta \|\pi_i(b_k)\|^2 \leq \|\pi_i(b_{k+1})\|^2 \). Thus the Lovász condition is implied by the second condition in Definition [4] restricted to consecutive pairs, i.e. \( \ell = k+1 \). \( \square \)

Lemma 3 For \( \delta \in (4^{-1/(n-1)},1] \), a \( \delta \)-DeepLLL reduced basis \( B \) is also \( \delta^{n-1} \)-PotLLL reduced.

Proof We proceed by contradiction. Assume that \( B \) is not \( \delta^{n-1} \)-PotLLL reduced, i.e. there exist \( 1 \leq k < \ell \leq n \) such that \( \delta^{n-1} \cdot \text{Pot}(B) > \text{Pot}(\sigma_{k,\ell}B) \). By Lemma [1] this is equivalent to

\[
\delta^{n-1} > \prod_{i=k}^{\ell} \frac{\|\pi_i(b_{\ell})\|^2}{\|\pi_i(b_i)\|^2} = \prod_{i=k}^{\ell-1} \frac{\|\pi_i(b_{\ell})\|^2}{\|\pi_i(b_i)\|^2}.
\]

It follows that there exists a \( j \in [k,\ell-1] \) such that \( \|\pi_j(b_{\ell})\|^2/\|\pi_j(b_{\ell})\|^2 < \delta^{(n-1)/(\ell-k)} \leq \delta \) which implies that \( B \) is not \( \delta \)-DeepLLL reduced. \( \square \)

3.1 High-Level Description

A high-level version of the algorithm is presented as Algorithm [1]. The algorithm is very similar to the classical LLL algorithm and the classical DeepLLL reduction by Schnorr and Euchner [SE94]. During its execution, the first \( \ell - 1 \) basis vectors are always \( \delta \)-PotLLL reduced (this guarantees termination of the algorithm). As opposed to classical LLL, and similar to DeepLLL, \( \ell \) might decrease by more than one. This happens precisely during deep insertions: in these cases, the \( \ell \)-th vector is not swapped with the \( (\ell-1) \)-th one, as in classical LLL, but with the \( k \)-th one for \( k < \ell - 1 \). In case \( k = \ell - 1 \), this equals the swapping of adjacent basis vectors as in classical LLL. The main difference of PotLLL and DeepLLL is the condition that controls insertion of a vector.
Algorithm 1: Potential LLL

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Input:} Basis \(B \in \mathbb{Z}^{n \times m}\), \(\delta \in (1/4, 1]\)
\State \textbf{Output:} A \(\delta\)-PotLLL reduced basis.
\State \(\ell \leftarrow 1\)
\While {\(\ell \leq n\)}
\State \(\text{Size-reduce}(B)\)
\State \(k \leftarrow \arg \min_{1 \leq j \leq \ell} \text{Pot}(\sigma_j B)\)
\If {\(\delta \cdot \text{Pot}(B) > \text{Pot}(\sigma_k B)\)}
\State \(B \leftarrow \sigma_k B\)
\State \(\ell \leftarrow k\)
\Else
\State \(\ell \leftarrow \ell + 1\)
\EndIf
\EndWhile
\State \Return \(B\)
\end{algorithmic}
\end{algorithm}

3.2 Detailed Description

There are two details to consider when implementing Algorithm 1. The first one is that since the basis vectors \(b_1, \ldots, b_{\ell-1}\) are already \(\delta\)-PotLLL reduced, they are in particular also size-reduced. Moreover, the basis vectors \(b_{\ell+1}, \ldots, b_n\) will be considered later again. So in line 3 of the algorithm it suffices to size-reduce \(b_\ell\) by \(b_1, \ldots, b_{\ell-1}\) as in classical LLL. Upon termination, when \(\ell = n+1\), the whole basis will be size-reduced.

Another thing to consider is the computation of the potentials of \(B\) and \(\sigma_j B\) for \(1 \leq j \leq \ell\) in line 4. Computing the potential of the basis is a rather slow operation. But we do not need to compute the potential itself, but only compare \(\text{Pot}(\sigma_k B)\) to \(\text{Pot}(B)\); by Lemma 1 this quotient can be efficiently computed. Define \(P_{k, \ell} := \text{Pot}(\sigma_k B)/\text{Pot}(B)\). The “if”-condition in line 5 will then change to \(\delta > P_{k, \ell}\), and the minimum in line 6 will change to \(\arg \min_{1 \leq j \leq \ell} P_{j, \ell}\). Using \(P_{\ell, \ell} = 1\) and

\[ P_{j, \ell} = \frac{\text{Pot}(\sigma_j B)}{\text{Pot}(B)} = P_{j+1, \ell} \cdot \frac{\|\pi_j(b_\ell)\|^2}{\|\pi_j(b_j)\|^2} = P_{j+1, \ell} \cdot \frac{\|b_\ell^*\|^2 + \sum_{i=j}^{\ell-1} \mu_{i, \ell} \|b_i^*\|^2}{\|b_j^*\|^2} \] (3.3)

for \(j < \ell\) (Lemma 1), we can quickly determine \(\arg \min_{1 \leq j \leq \ell} P_{j, \ell}\) and check whether \(\delta > P_{k, \ell}\) if \(j\) minimizes \(P_{j, \ell}\).

A detailed version of Algorithm 1 with these steps filled in is described as Algorithm 2. On line 7 of Algorithm 2 \(P_{j, \ell}\) is iteratively computed as in Equation (3.3). Clearly, the algorithm could be further improved by iteratively computing \(\|\pi_j(b_{j})\|^2\) from \(\|\pi_{j+1}(b_{j})\|^2\). Depending on the implementation of the Gram-Schmidt orthogonalization, this might already have been computed and stored. For example, when using the Gram-Schmidt orthogonalization as described in Figure 4 of [NS05], then \(\|\pi_j(b_j)\|^2 = s_j - 1\) after computation of \(\|b_j^*\|^2\) and \(\mu_{i,j}\) for \(1 \leq j < \ell\).

3.3 Complexity Analysis

Here we show that the number of operations in the PotLLL algorithm are bounded polynomially in the dimension \(n\) and the logarithm of the input size. We present the runtime for Algorithm 2.

**Proposition 1** Let \(\delta \in (1/4, 1]\) and \(C = \max_{1 \leq i \leq n} \|b_i\|^2\). Then Algorithm 2 performs \(O(n^3 \log_{1/\delta}(C))\) iterations of the while loop in line 4 and a total of \(O(n \min^4 \log_{1/\delta}(C))\) arithmetic operations.

**Proof** Let us start by upper bounding the potential \(I\) of the input basis with respect to \(C\). Let \(d_j := \text{vol}(\mathcal{L}(b_1, \ldots, b_j))^2 = \prod_{i=1}^{j} \|b_i^*\|^2\) for \(j = 1, \ldots, n\). Recall that \(\|b_i^*\|^2 \leq \|b_i\|^2 \leq C\) for \(i = 1, \ldots, n\)
Algorithm 2: Potential LLL, detailed version

| Input: | Basis $B \in \mathbb{Z}^{n \times m}$, $\delta \in (1/4, 1]$. |
| Output: | A $\delta$-PotLLL reduced basis. |

1 $\ell \leftarrow 1$
2 while $\ell \leq n$
3 
4 
5 $P \leftarrow 1$, $P_{\min} \leftarrow 1$, $k \leftarrow 1$
6 for $j = \ell - 1$ down to 1 do
7 
8 if $P < P_{\min}$ then
9 
10 
11 end
12 end
13 if $\delta > P_{\min}$ then
14 $B \leftarrow \sigma_{k,j}B$
15 
16 else
17 $\ell \leftarrow \ell + 1$
18 end
19 end
20 return $B$

and hence $d_j < C^j$. Consequently we have the following upper bound on the potential

$$I = \prod_{j=1}^{n-1} d_j \cdot \text{vol}(\mathcal{L}) \leq \prod_{j=1}^{n-1} C^j \cdot \text{vol}(\mathcal{L}) \leq C^{\frac{n(n-1)}{2}} \cdot \text{vol}(\mathcal{L}).$$

(3.4)

Now, by a standard argument, we show that the number of iterations of the while loop is bounded by $O(n^3 \log_{1/\delta}(C))$. In each iteration, either the iteration counter $\ell$ is increased by 1, or an insertion takes place and $\ell$ is decreased by at most $n - 1$. In the insertion case, the potential is decreased by a factor at least $\delta$. So after $N$ swaps the potential $I_N$ satisfies $I \geq (1/\delta)^N I_N \geq (1/\delta)^N \cdot \text{vol}(\mathcal{L})$ using the fact that $I_N \geq \text{vol}(\mathcal{L})$. Consequently the number of swaps $N$ is bounded by $N \leq \log_{1/\delta}(I/\text{vol}(\mathcal{L}))$. By Equation (3.4) we get that $N \leq \log_{1/\delta}(C^m(n-1)/2)$. Now note that the number $M$ of iterations where $\ell$ is increased by 1 is at most $M \leq (n - 1) \cdot N + n$. This shows that the number of iterations is bounded by $O(n^3 \log_{1/\delta}(C))$.

Next we show that the number of operations performed in each iteration of the while loop is dominated by $O(nm)$ operations. Size-reduction (line 3) and the first update step (line 4) can be done in $O(nm)$ steps. The for-loop consists of $O(n)$ iterations where the most expensive operation is the update of $P$ in line 7. Therefore the loop requires $O(nm)$ arithmetic operations. Insertion can be done in $O(n)$ operations, whereas the second update in line 14 requires again $O(nm)$ operations.

It follows that each iteration costs at most $O(nm^4 \log(C))$ arithmetic operations. This shows that in total the algorithm performs $O(mn^4 \log(C))$ operations.

3.4 Worst-Case Behavior

For $\delta = 1$, there exist so called critical bases which are $\delta$-LLL reduced bases and whose Hermite factor reaches the worst case bound in [22, Sch94]. These bases can be adapted to form a DeepLLL reduced basis where the first vector reaches the worst case bound in [22].
Proposition 2 For $\alpha = \sqrt{3/4}$, the rows of $B = A_n(\alpha)$ (see below) define a $\delta$-DeepLLL reduced basis with $\delta = 1$ and $\|b_1\|^2 = \frac{1}{\alpha^{n-1/2}} \text{vol}(\mathcal{L}(A_n))^{1/n}$.

$A_n(\alpha) := \begin{pmatrix}
1 & 0 & \cdots & \cdots & 0 \\
\frac{1}{2} & \alpha & \vdots & \vdots & \vdots \\
\vdots & \frac{1}{2} & \alpha^2 & \vdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & \cdots & \cdots & \cdots \\
\frac{1}{2} & \frac{1}{2} & \alpha^{n-2} & \cdots & \alpha^{n-1}
\end{pmatrix}$

(3.5)

Proof From the diagonal form of $A_n$ it is easy to see that $\text{vol}(\mathcal{L}) = \det(A_n) = \alpha^{n(n-1)/2}$. Hence $\|b_1\|^2 = 1 = \frac{1}{\alpha^{n-1/2}} \text{vol}(\mathcal{L})$. It remains to show that $A_n$ is DeepLLL reduced. Note that $B^*$ is a diagonal matrix with the same entries on the diagonal as $B$. Note that it is size reduced as for all $1 \leq j < i \leq n$ we have $\mu_{i,j} = \frac{\langle b_i, b^*_j \rangle}{\langle b^*_j, b^*_j \rangle} = \frac{1}{2} \frac{\alpha^{2(j-1)}}{\alpha^{2(i-1)}} = \frac{1}{2}$. Further, using that $\pi_j(b_i) = b^*_i + \sum_{\ell=j}^{i-1} \mu_{i,\ell} b^*_\ell$, we have that

$$\|\pi_j(b_i)\|^2 = \alpha^{2(i-1)} + \frac{1}{4} \sum_{\ell=j}^{i-1} \alpha^{2(\ell-1)} = \alpha^{2(j-1)} \left( \frac{1}{4} \sum_{\ell=0}^{i-1-j} \alpha^{2\ell} + \alpha^{2(i-j)} \right).$$

As for $\alpha = \sqrt{3/4}$, we have that $\frac{1}{4} \sum_{\ell=0}^{i-1-j} \alpha^{2\ell} + \alpha^{2(i-j)} = 1$, and hence $\|\pi_j(b_i)\|^2 = \alpha^{2(j-1)} = \|\pi_j(b_i)\|^2$. Therefore, the norms of the projections for fixed $j$ are all equal, and $A_n(\alpha)$ is $\delta$-PotLLL-reduced with $\delta = 1$. \qed

Using Lemma 3 we obtain:

Corollary 1 For $\alpha = \sqrt{3/4}$, the rows of $A_n(\alpha)$ define a $\delta$-PotLLL reduced basis with $\delta = 1$ and $\|b_1\|^2 = \frac{1}{\alpha^{n-1/2}} \text{vol}(\mathcal{L}(A_n))^{1/n}$. \qed

4 Experimental Results

Extensive experiments have been made to examine how the classical LLL reduction algorithm performs in practice [NS06, GN08]. We ran some experiments to compare our PotLLL algorithm to our implementations of LLL, DeepLLL, and BKZ.

4.1 Setting

We run the following algorithms, each with the standard reduction parameter $\delta = 0.99$:

1. classical LLL,
2. PotLLL,
3. DeepLLL with blocksize $\beta = 5$ and $\beta = 10$ (the latter up to dimension 240 only),
4. BKZ with blocksize 5 (BKZ-5) and 10 (BKZ-10).

The implementations all use the same arithmetic back-end. Integer arithmetic is done using GMP, and Gram-Schmidt arithmetic is done as described in [NS05, Figures 4 and 5]. As floating point types, long double (x64 extended precision format, 80 bit representation) and MPFR arbitrary
precision floating point numbers are used with a precision as described in [NS05]. The implementations of DeepLLL and BKZ follow the classical description in [SE94]. PotLLL was implemented as described in Algorithm 2 (page 9). Our implementation will be made publicly available.

We ran experiments in dimensions 40 to 300, considering the dimensions which are multiples of 10. For DeepLLL with \( \beta = 10 \) we ran experiments only up to dimension 240. In each dimension, we considered 50 random lattices. More precisely, we used the lattices of seed 0 to 49 from the SVP Challenge\(^{1}\). For each lattice, we used two bases: the original basis and a 0.75-LLL reduced basis.

All experiments were run on Intel\textsuperscript{®} Xeon\textsuperscript{®} X7550 CPUs at 2 GHz on a shared memory machine. For dimensions 40 up to 160, we used long double arithmetic, and for dimensions 160 up to 300, we used MPFR. In dimension 160, we did the experiments both using long double and MPFR arithmetic. The reduced lattices did not differ. In dimension 170, floating point errors prevented the long double arithmetic variant to complete on some of the lattices.

### 4.2 Results

For each run, we recorded the length of the shortest vector as well as the required CPU time for the reduction. Our main interest lies in the \( n \)-th root of the Hermite factor \( \frac{\|b_1\|}{\text{vol}(\mathcal{L})^{1/n}} \), where \( b_1 \) is the shortest vector of the basis of \( \mathcal{L} \) returned. Figure 1 (see pages 10–12 for all figures) compares the average \( n \)-th root of the Hermite factor and average logarithmic running time of the algorithms for all dimensions. The graphs also show confidence intervals for the average value with a confidence level of 99.9%.

As one can see, there is a clear hierarchy with respect to the achieved Hermite factor. Our PotLLL performs better than BKZ-5, though worse than DeepLLL with \( \beta = 5 \) and BKZ-10, which in turn perform worse than DeepLLL with \( \beta = 10 \). The behavior for preprocessed bases and bases in Hermite normal form is very similar. We collected the average \( n \)-th root Hermite factors \( \|b_1\|^{1/n} \cdot \text{vol}(\mathcal{L})^{-1/n} \) in Table 1 and compared them to the worst-case bound in Equation (2.2). Our data for LLL is similar to the one in [NS06] and [GN08, Table 1]. However, we do not see convergence of the \( n \)-th root Hermite factors in our experiments, as they are still increasing even in high dimensions \( n > 200 \).

<table>
<thead>
<tr>
<th>Dimension</th>
<th>( n = 100 )</th>
<th>( n = 160 )</th>
<th>( n = 220 )</th>
<th>( n = 300 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case bound (proven)</td>
<td>( \approx 1.0774 )</td>
<td>( \approx 1.0777 )</td>
<td>( \approx 1.0778 )</td>
<td>( \approx 1.0779 )</td>
</tr>
<tr>
<td>Empirical 0.99-LLL</td>
<td>1.0187</td>
<td>1.0201</td>
<td>1.0207</td>
<td>1.0212</td>
</tr>
<tr>
<td>Empirical 0.99-BKZ-5</td>
<td>1.0154</td>
<td>1.0158</td>
<td>1.0161</td>
<td>1.0163</td>
</tr>
<tr>
<td>Empirical 0.99-PotLLL</td>
<td>1.0146</td>
<td>1.0150</td>
<td>1.0151</td>
<td>1.0153</td>
</tr>
<tr>
<td>Empirical 0.99-DeepLLL with ( \beta = 5 )</td>
<td>1.0138</td>
<td>1.0142</td>
<td>1.0147</td>
<td>1.0150</td>
</tr>
<tr>
<td>Empirical 0.99-BKZ-10</td>
<td>1.0140</td>
<td>1.0143</td>
<td>1.0144</td>
<td>1.0145</td>
</tr>
<tr>
<td>Empirical 0.99-DeepLLL with ( \beta = 10 )</td>
<td>1.0128</td>
<td>1.0132</td>
<td>1.0135</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 1: Worst case bound and average case estimate for \( \delta \)-LLL reduction, \( \delta \)-DeepLLL reduction, \( \delta \)-PotLLL reduction and \( \delta \)-BKZ reduction of the \( n \)-th root Hermite factor \( \|b_1\|^{1/n} \cdot \text{vol}(\mathcal{L})^{-1/n} \). The entries are sorted in descending order with respect to the observed Hermite factors.

For the running time comparison, Figure 1b shows that the observed order is similar to the order induced by the Hermite factors. LLL is fastest, followed by BKZ-5 and PotLLL, then by BKZ-10 and DeepLLL with \( \beta = 5 \), and finally there is DeepLLL with \( \beta = 10 \). The running time

\[^{1}\text{http://www.latticechallenge.org/svp-challenge}\]
of PotLLL and BKZ-5 is very close to each other for higher dimensions, while PotLLL is clearly slower for smaller dimensions. While Figure 1b shows that BKZ-5 is usually slightly faster than PotLLL and BKZ-10 slightly faster than DeepLLL with $\beta = 5$, the behavior is more interesting if one considers preprocessed and non-preprocessed bases separately. We do this in Figures 2 and 3. Recall that the unprocessed bases are bases in Hermite normal form, and the processed bases are the same bases run through 0.75-LLL.

In Figures 2 we compare the behavior for unprocessed bases in Hermite normal form. Every line connecting bullets corresponds to the behavior of one algorithm for different dimensions. Again, the box surrounding a bullet is a confidence interval with confidence level 99.9%. The shaded regions show which Hermite factors can be achieved in every dimension by these algorithms. Algorithms on the border of the region are optimal for their Hermite factor: none of the other algorithms in this list produces a better average Hermite factor in less time. In Figure 2a one can see that BKZ-5 produces worse output slower than PotLLL up to dimension 160. Also, BKZ-10 is inferior to DeepLLL with $\beta = 5$ as it is both slower and produces worse Hermite factors. As the dimension increases, the difference in running time becomes less and less. In fact, for dimension 180 and larger, BKZ-10 becomes faster than DeepLLL with $\beta = 5$ (Figure 2b).

On the other hand, for preprocessed bases, the behavior is different, as Figure 3 shows. Here, BKZ-5 is clearly faster than PotLLL and BKZ-10 clearly faster than DeepLLL with $\beta = 5$. In fact, for dimensions 60, 80 and 100, PotLLL is slower than BKZ-10 while producing worse output (Figure 3a). For higher dimensions, PotLLL is again faster than BKZ-10 (Figure 3b), though not substantially. Therefore, for preprocessed bases, it seems that BKZ-10 is more useful than PotLLL and DeepLLL with $\beta = 5$

Finally, we tried adding PotLLL to the fplll library\footnote{\url{http://xpujol.net/fplll/}} first experiments show that the relation between LLL and PotLLL in fplll is very similar to the relation between LLL and PotLLL in our implementation. A more elaborate comparison, in particular with fplll’s BKZ implementation, is still in progress.

5 Conclusion

We present a first provable polynomial time variant of Schnorr and Euchner’s DeepLLL. While the provable bounds are not better than for classical LLL – in fact, for reduction parameter $\delta = 1$, the existence of critical bases shows that better bounds do not exist – the practical behavior is much better than for classical LLL. We see that the $\sqrt[3]{n}$-th root Hermite factor of an $n$-dimensional basis output by PotLLL in average does not exceed $1.0153^n$ for $n \leq 300$.

For unprocessed random bases in Hermite normal form, PotLLL even outperforms BKZ-5. Our experiments also show that for such bases, DeepLLL with $\beta = 5$ outperforms BKZ-10. On the other hand, for bases which are already reasonably preprocessed, for example by applying 0.75-LLL to a basis in Hermite normal form, our algorithm is only slightly faster and sometimes even slower than BKZ-10, while producing longer vectors.

It is likely that the improvements of the $L^2$ algorithm\cite{NS06} and the $L^1$ algorithm\cite{NSVT11} can be used to improve the runtime of our PotLLL algorithm, in order to achieve faster runtime. We leave this for future work.

Moreover, deep insertions can be used together with BKZ as well. In particular, potential minimizing deep insertions can be used. We added classical deep insertions and potential minimizing deep insertions to BKZ. First experiments up to dimension 120 suggest that with regard to the output quality, BKZ-5 with potential minimizing deep insertions is better than PotLLL, but worse than BKZ-5 with classical deep insertions, which in turn comes close to BKZ-10. BKZ-10 with potential minimizing deep insertions is close to DeepLLL with $\beta = 5$, and BKZ-10 with classical deep insertions close to DeepLLL with $\beta = 10$. For dimensions around 100, the speed of similarly performing algorithms also behaves similarly.
Fig. 1: Overview of performance of the algorithms for dimensions \( n \) (x axis) from 40 to 300 (using MPFR for \( n \geq 160 \)).

Acknowledgements This work was supported by CASED (http://www.cased.de). Michael Schneider is supported by project BU 630/23-1 of the German Research Foundation (DFG). Urs Wagner and Felix Fontein are supported by SNF grant no. 132256.

References


Fig. 2: Comparison of \( n \)-th root Hermite factor (y axis) vs. running times (x axis) for both arithmetics for the original bases.

(a) **long double** arithmetic. The highlighted areas represent dimensions 40, 80, 120 and 160.

(b) MPFR arithmetic. The highlighted areas represent dimensions 180, 220, 260 and 300.


Fig. 3: Comparison of $n$-th root Hermite factor (y axis) vs. running times (x axis) for both arithmetics for preprocessed bases (0.75-LLL-reduced bases).

(a) long double arithmetic. The highlighted areas represent dimensions 40, 80, 120 and 160.

(b) MPFR arithmetic. The highlighted areas represent dimensions 180, 220, 260 and 300.


On the normality of \( p \)-ary bent functions

Ayça Çeşmelioğlu · Wilfried Meidl · Alexander Pott

Received: date / Accepted: date

Abstract In this work, the normality of bent functions in odd characteristic is analysed. It turns out that differently to Boolean bent functions, many - also quadratic - bent functions in odd characteristic and even dimension are not normal. It is shown that regular Coulter-Matthews bent functions are normal.

Keywords Bent Functions · Normality · Coulter-Matthews bent functions

1 Introduction

For a prime \( p \), let \( f \) be a function from an \( n \)-dimensional vector space \( V_n \) over \( \mathbb{F}_p \) to \( \mathbb{F}_p \). The Walsh transform of \( f \) is then defined to be the complex valued function \( \hat{f} \) on \( V_n \)

\[
\hat{f}(b) = \sum_{x \in V_n} \epsilon_p^{f(x) - \langle b, x \rangle}
\]

where \( \epsilon_p = e^{2\pi i/p} \) and \( \langle b, x \rangle \) denotes a (nondegenerate) inner product on \( V_n \). The classical frameworks are \( V_n = \mathbb{F}_p^n \) and \( \langle b, x \rangle \) is the conventional dot product denoted by ",", and \( V_n = \mathbb{F}_p^n \) and \( \langle b, x \rangle = \text{Tr}_n(bx) \), where \( \text{Tr}_n(z) \) denotes the absolute trace of \( z \in \mathbb{F}_p^n \). In this contribution we will consider examples in both frameworks, but general definitions and results will be formulated in the framework of \( V_n = \mathbb{F}_p^n \).

A. Çeşmelioğlu is supported by Tübitak BİDEB 2219 Scholarship Programme. W.Meidl is supported by Tübitak Project no.111T234.

Ayça Çeşmelioğlu · Alexander Pott
Otto-von-Guericke-University, Faculty of Mathematics, 39106 Magdeburg, Germany
E-mail: cesmelio@ovgu.de, alexander.pott@ovgu.de

Wilfried Meidl
Sabanci University, MDBF, Orhaneli, Tuzla, 34956 İstanbul, Turkey
E-mail: wmeidl@sabanciuniv.edu
The function $f$ is called a **bent function** if $|\widehat{f}(b)| = p^{n/2}$ for all $b \in \mathbb{F}_p^n$. If $|\widehat{f}(b)| \in \{0, p^{(n+1)/2}\}$ for all $b \in \mathbb{F}_p^n$, then we call $f$ **near-bent** (for $p = 2$ the term semi-bent is common), and more generally $f$ is called $s$-plateaued for an integer $0 \leq s \leq n$ if $|\widehat{f}(b)| \in \{0, p^{(n+s)/2}\}$ for all $b \in \mathbb{F}_p^n$. We remark that for $p = 2$ the Walsh transform yields an integer. Hence if $f$ is $s$-plateaued, then $n$ and $s$ must have the same parity. In particular, binary bent functions only exist for $n$ even. For odd $p$, bent functions exist for $n$ even and for $n$ odd.

For the Walsh coefficient $\widehat{f}(b)$ we always have (cf. [8])

$$p^{-n/2} \widehat{f}(b) = \begin{cases} \pm f^*(b) & : \text{n even or } n \text{ odd and } p \equiv 1 \mod 4 \\ \pm i f^*(b) & : \text{n odd and } p \equiv 3 \mod 4, \end{cases} \quad (1)$$

where $f^*$ is a function from $\mathbb{F}_p^n$ to $\mathbb{F}_p$. A bent function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ is called **regular** if for all $b \in \mathbb{F}_p^n$,

$$p^{-n/2} \widehat{f}(b) = \varepsilon f^*(b).$$

When $p = 2$, a bent function is trivially regular, and as can be seen from (1), for $p > 2$ a regular bent function can only exist for even $n$ and for odd $n$ when $p \equiv 1 \mod 4$. A function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ is called **weakly regular** if, for all $b \in \mathbb{F}_p^n$, we have

$$p^{-n/2} \widehat{f}(b) = \zeta \varepsilon f^*(b)$$

for some complex number $\zeta$ with $|\zeta| = 1$, otherwise it is called **not weakly regular**. By (1), $\zeta$ can only be $\pm 1$ or $\pm i$. Note that regular implies weakly regular.

The classical example for a bent function is the **Maiorana-McFarland** bent function from $\mathbb{F}_p^m \times \mathbb{F}_p^m = \mathbb{F}_p^{2m}$ to $\mathbb{F}_p$ defined by

$$f(x, y) = x \cdot \pi(y) + \sigma(y)$$

for a permutation $\pi$ of $\mathbb{F}_p^m$ and an arbitrary function $\sigma : \mathbb{F}_p^m \to \mathbb{F}_p$. We remark that the condition that $\pi$ is a permutation is necessary and sufficient for $f$ being bent. The Maiorana-McFarland function is always a regular bent function. Moreover $f(x, \pi^{-1}(0)) = \sigma(\pi^{-1}(0))$ is constant for all $x \in \mathbb{F}_p^m$, hence a Maiorana-McFarland function in an example of a normal function, which is defined as follows. For an even integer $n = 2m$, a function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ is called **normal** if there is an affine subspace of dimension $m = n/2$ on which the function is constant, $f$ is called **weakly normal** if there is an affine subspace of dimension $m = n/2$ on which the function is affine, see [11]. The notion of normal Boolean functions was introduced in [6]. By counting arguments one can show that nearly all Boolean functions are non-normal, however almost all known Boolean bent functions are normal, see [11].
2 Normality of $p$-ary bent functions

Normality for Boolean bent functions was investigated in the articles [2,3,5,11]. Recently, in [10] a $p$-ary bent function has been shown to be normal. In this section we further investigate normality for $p$-ary bent functions.

As easily observed, normality of functions from $\mathbb{F}_p^n$ to $\mathbb{F}_p$ is invariant under coordinate trasformation. However, the addition of an affine function alters a normal to a weakly normal function. Hence normality is not preserved under EA-equivalence transformations. Consequently in relation to bentness, which is invariant under EA-equivalence, in view of the following lemma the concept of weak normality is more natural (see also [5]).

**Lemma 1** A function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ is weakly normal if and only if there exists an $a \in \mathbb{F}_p^n$ such that $f(x) - a \cdot x$ is normal.

*Proof* For an even integer $n = 2m$ let $f : \mathbb{F}_p^n \to \mathbb{F}_p$ be weakly normal. Then there exists a subspace $E$ of dimension $m$ such that $f(x)$ is affine on $b + E$ for some $b \in \mathbb{F}_p^n$. Assume that $f(x) = d \cdot x + c, c \in \mathbb{F}_p$, $d \in \mathbb{F}_p^*$, on $b + E$ and choose any $a \in d + E^\perp$, and consider the function $f(x) - a \cdot x$. On $b + E$, the values of the function are given as $f(x) - a \cdot x = (d - a) \cdot x + c = (d - a) \cdot b + c$ since $d - a \in E^\perp$. Hence $f(x) - a \cdot x$ is normal. The converse follows similarly. 

The following theorem establishes a relationship between regularity and normality of $p$-ary bent functions, and it analyses the behaviour of $p$-ary (weakly) normal functions on the cosets of the subspace $E$ wherever $f$ is affine.

**Theorem 1** For $n = 2m$ let $f : \mathbb{F}_p^n \to \mathbb{F}_p$ be a bent function.

(i) If $f$ is weakly regular but not regular, then $f$ is not (weakly) normal.

(ii) If $f$ is normal, hence constant on $E + b$ for an $m$-dimensional subspace $E$ and $b \in \mathbb{F}_p^n$, then $f$ is balanced on the remaining cosets. The dual $f^*$ of $f$ is (weakly) normal.

*Proof* Let $E$ be an arbitrary subspace of $\mathbb{F}_p^n$, let $b \in \mathbb{F}_p^n$ and let $E^\perp$ be the orthogonal complement of $E$ in $\mathbb{F}_p^n$. Then

$$
\sum_{u \in E^\perp} \epsilon_{p}^{b \cdot u} f(u) = \sum_{u \in E^\perp} \epsilon_{p}^{b \cdot u} \sum_{x \in \mathbb{F}_p^n} \epsilon_{p}^{f(x) - u \cdot x} = \sum_{x \in \mathbb{F}_p^n} \epsilon_{p}^{f(x)} \sum_{u \in E^\perp} \epsilon_{p}^{(b - x) \cdot u} \sum_{x \in b + E} \epsilon_{p}^{f(x)}.
$$

Let $f : \mathbb{F}_p^n \to \mathbb{F}_p, n = 2m$, be a normal bent function, let $E$ be an $m$-dimensional subspace of $\mathbb{F}_p^n$ and $b \in \mathbb{F}_p^n$ such that $f(x) = c \in \mathbb{F}_p$ for all $x \in b + E$.

Since $n$ is even, the possible Walsh transform values are $\hat{f}(u) = \pm p^m \epsilon_{p}^{f(x) u}$ for any $u \in \mathbb{F}_p^n$. With Equation (2) we then obtain

$$
p^m \sum_{u \in E^\perp} (-1)^{\hat{f}(u) + b \cdot u} = p^{m} \epsilon_{p}^{c}, \quad \text{i.e.} \quad \sum_{u \in E^\perp} (-1)^{\hat{f}(u) + b \cdot u} = p^{m} \epsilon_{p}^{c}.
$$
where $j_u \in \{0, 1\}$ for all $u \in E^\perp$. Consequently, we require that for all $u \in E^\perp$ we have $j_u = 0$ and $f^*(u) + b \cdot u = c$. The first condition implies that $\tilde{f}(u) = p^m \epsilon_p f^*(u)$ for all $u \in E^\perp$. Hence the normal bent function $f$ must be either regular or not weakly regular with Walsh coefficients with a positive sign on $E^\perp$. Consequently, we require that for all $u \in E^\perp$ we have $j_u = 0$ and $f^*(u) + b \cdot u = c$. The first condition implies that $\tilde{f}(u) = p^m \epsilon_p f^*(u)$ for all $u \in E^\perp$. Hence the normal bent function $f$ must be either regular or not weakly regular with Walsh coefficients with a positive sign on $E^\perp$. Since for a weakly regular (but not regular) bent function $f(x)$ also $f(x) - a \cdot x$ is weakly regular (but not regular), by Lemma 1 a weakly regular (but not regular) bent function cannot be weakly normal as well, which finishes the proof for (i).

The second condition, $f^*(u) = c - b \cdot u$ for all $u \in E^\perp$, implies that the dual $f^*$ of a normal bent function $f$ is weakly normal.

Let $b' \in F_2^n$ and $b' \notin b + E$. Then with Equation (2) we get
\[
\sum_{u \in E^\perp} b'^u p^m \epsilon_p f^*(u) = p^m \sum_{x \in b' + E} \epsilon_p f(x).
\]

Consequently,
\[
\sum_{x \in b' + E} \epsilon_p f(x) = \sum_{u \in E^\perp} \epsilon_p f^*(u) + b' \cdot u = \epsilon_p \sum_{u \in E^\perp} \epsilon_p (b' - b) \cdot u = 0.
\]
The last equality follows since $f^*(u) = c - b \cdot u$ on $E^\perp$ and $b' - b \notin E$. Hence the function $f$ is balanced on each coset of $E$ in $F_2^n$ except $b + E$, which finishes the proof.

We remark that weakly normal functions $f$ in dimension $n$ are affine on an $n/2$-dimensional subspace $E$, whereas the completed Maiorana-McFarland class (the set of all bent functions EA-equivalent to a Maiorana-McFarland function) is characterized by the much stronger condition that $f$ is also affine on every coset of $E$, see [2, Lemma 33]. Hence the completed Maiorana-McFarland class is trivially weakly normal. Using that there exist exactly two EA-inequivalent classes of quadratic bent functions, one with solely regular bent functions and one with weakly regular bent functions only, Theorem 1 confirms results in [1] according to which not all quadratic bent functions in odd characteristic are in the completed Maiorana-McFarland class. More generally, we can show the following theorem.

**Theorem 2** Let $f$ be a quadratic bent function in odd characteristic and dimension $n$. Then
- $f$ is in the completed Maiorana-McFarland class if $n$ is even and $f$ is regular,
- $f$ is affine on all cosets of an $s = (n - 2)/2$-dimensional subspace $E$ if $n > 2$ is even and $f$ is weakly regular,
- $f$ is affine on all cosets of an $s = (n - 1)/2$-dimensional subspace $E$ if $n > 1$ is odd.

Combining Theorem 2 with [8, Corollary 6] on the signs of the Walsh coefficients for quadratic bent monomials, we obtain the subsequent corollary, which generalizes Theorem 4 in [1].
Corollary 1 For an even integer $n$, the monomial bent function $f(x) = \text{Tr}_n(ax^{p^j+1})$, $1 \leq j \leq n$, $\frac{n}{\gcd(n, p^j)}$ odd, from $\mathbb{F}_{p^n}$ to $\mathbb{F}_p$ is in the completed Maiorana-McFarland class if and only if

- $p \equiv 1 \mod 4$ and $a$ is a nonsquare in $\mathbb{F}_{p^n}$, or
- $p \equiv 3 \mod 4$, $n \equiv 2 \mod 4$, and $a$ is a square in $\mathbb{F}_{p^n}$, or
- $p \equiv 3 \mod 4$, $n \equiv 0 \mod 4$, and $a$ is a nonsquare in $\mathbb{F}_{p^n}$.

By Theorem 1, not weakly regular bent functions, from which one may expect a more chaotic behaviour than from weakly regular bent functions, still may be normal. In the following example we show the normality of a not weakly regular bent function presented in [9].

Example 1 Let $\omega$ be a root of the irreducible polynomial $x^4 + x + 2 \in \mathbb{F}_3[x]$, which is a primitive element of $\mathbb{F}_{3^4}$. Then $f : \mathbb{F}_{3^4} \rightarrow \mathbb{F}_3$ given by $f(x) = \text{Tr}_4(\omega^{10}x^{22} + x^4)$ is not weakly regular bent, see [9]. With Magma we observe that $f(x) = 0$ for all $x$ in the 2-dimensional subspace $E := \text{span}\{\omega, \omega^3 + \omega^2\}$. Hence $f$ is normal. From Theorem 1 and its proof we know that then the dual $f^*$ is affine on the orthogonal complement $E^\perp = \text{span}\{1, \omega^2 + 2\omega\}$ (with respect to $\text{Tr}_n(xy)$) and $f^*$ is weakly normal. Looking at the Walsh coefficients we observe that $\hat{f}(u) = 9$ for all $u \in E^\perp$, hence $f^*(u) = 0$ for all $u \in E^\perp$ and $f^*$ is normal as well.

Remark 1 Differently to (weakly) regular bent functions, the dual of a not weakly regular bent function need not be bent. The function $f(x) = \text{Tr}_4(\omega^{10}x^{22} + x^4)$ is an example of a not weakly regular bent function for which the dual is not bent, see [4].

The most famous non-quadratic bent functions in odd characteristic are the coordinate functions $f_\alpha : \mathbb{F}_{3^n} \rightarrow \mathbb{F}_3$

$$f_\alpha(x) = \text{Tr}_n(\alpha x^{3^{\frac{n+1}{2}}}), \quad \alpha \in \mathbb{F}_{3^n}^*,$$

of the Coulter-Matthews perfect nonlinear function $f : \mathbb{F}_{3^n} \rightarrow \mathbb{F}_3$

$$f(x) = x^{3^{\frac{n-1}{2}}}, \quad k \text{ odd and } \gcd(n, k) = 1,$$

which is the only known non-quadratic perfect nonlinear function. We close this section with an analysis of the normality of this family of bent functions.

 Proposition 1 Let $n = 2m$ with $m \geq 1$. Then for each $\alpha \in \mathbb{F}_{3^n}^*$, the Walsh transform $\hat{f}_\alpha$ of the weakly regular bent function $f_\alpha(x) = \text{Tr}_n(\alpha x^{2^{\frac{n+1}{2}}})$, $k$ is odd, $\gcd(n, k) = 1$, satisfies

$$\hat{f}_\alpha(\beta) = \begin{cases} -\eta(\alpha)3^m e_{3^n}(\beta) & n \equiv 0 \mod 4, \\ \eta(\alpha)3^m e_{3^n}(\beta) & n \equiv 2 \mod 4, \end{cases}$$

for all $\beta \in \mathbb{F}_{3^n}$, where $\eta$ represents the quadratic character on $\mathbb{F}_{3^n}$.
Proof For each $\alpha \in \mathbb{F}_3^n$, and $\beta \in \mathbb{F}_3^n$, with Lemma 3 of [7] we have
\[ \hat{f}_\alpha(\beta) = \epsilon_{\alpha,\beta} 3^m \] and $\epsilon_{\alpha,0} \in \{-1,1\}$ and $\epsilon_{\alpha,\beta} \epsilon_{\alpha,0} \in \{1, \epsilon_3, \epsilon_3^2\}$.

Hence for each $\alpha \in \mathbb{F}_3^n$, one needs to find $\epsilon_{\alpha,0}$ for determining when $f_\alpha$ is regular bent or not. With the following argument in the proof of Lemma 3 of [7], the Walsh transform of $f_\alpha$ is reduced to the quadratic case:
\[ \hat{f}_\alpha(0) = \sum_{x \in \mathbb{F}_3^n} \epsilon_3^{\text{Tr}_n(\alpha x^{\frac{3^{j+1}}{2}})} = \sum_{x \in \mathbb{F}_3^n} \epsilon_3^{\text{Tr}_n(\alpha x^2)} \]

since $\gcd(\frac{3^{j+1}}{2}, 3^n - 1) = 2$. Therefore
\[ \hat{f}_\alpha(0) = \eta(\alpha)(-1)^{n-1}i^{n^2} \]

The last step follows from Theorem 5.33, 5.15 in [12], i.e. the result on Gaussian sums.

Remark 2 For even $n$, half of the coordinate functions of the Coulter-Matthews polynomial are regular and the rest are weakly regular (but not regular) bent. The conditions in the previous proposition explicitly describe the Walsh transform values. Namely, the bent functions $f_\alpha(x) = \text{Tr}_n(\alpha x^{\frac{3^{j+1}}{2}})$ are regular bent if $n \equiv 0 \mod 4$ and $\alpha$ is a nonsquare in $\mathbb{F}_{3^n}$ or $n \equiv 2 \mod 4$ and $\alpha$ is a square in $\mathbb{F}_{3^n}$ and weakly regular (but not regular) bent if $n \equiv 0 \mod 4$ and $\alpha$ is a square in $\mathbb{F}_{3^n}$ or $n \equiv 2 \mod 4$ and $\alpha$ is a nonsquare in $\mathbb{F}_{3^n}$.

Theorem 3 The regular Coulter-Matthews bent functions are normal.

Proof Assume $\omega \in \mathbb{F}_{3^n}$ is a primitive element of $\mathbb{F}_{3^n}$. Let $n = 2m$ and $f_\epsilon(x) = \text{Tr}_n(\omega^\epsilon x^{\frac{3^j+1}{2}})$ be the Coulter-Matthews bent function from $\mathbb{F}_{3^n}$ to $\mathbb{F}_3$ where $0 \leq \epsilon \leq 3^n - 2$ and $\epsilon$ is even if $m$ is odd and $\epsilon$ odd if $m$ is even.

Any $x \in \mathbb{F}_{3^n}$ can be represented in the form $x = \omega^{(3^m+1)j}z, 0 \leq j \leq 3^m$. This representation corresponds to the partition $\mathbb{F}_{3^n} = \bigcup_{j=0}^{3^m} \omega^j \mathbb{F}_{3^m}$.

Then,
\[ f_\epsilon(x) = \text{Tr}_n(\omega^\epsilon x^{\frac{3^j+1}{2}}) = \text{Tr}_n(\omega^\epsilon x^{\frac{3^j+1}{2}} \omega^{(3^m+1)j} \omega^{(3^m+1)\frac{3^j+1}{2}}) \]

We want to find $j, 0 \leq j \leq 3^m$, such that $\text{Tr}_n(\omega^{\epsilon+\frac{3^j+1}{2}} \omega^{(3^m+1)\frac{3^j+1}{2}}) = 0$ for all $0 \leq \epsilon \leq 3^n - 2$. For a fixed $j$, let $\alpha = \omega^{\epsilon+\frac{3^j+1}{2}}$ and $z = \omega^{(3^m+1)\frac{3^j+1}{2}}$. Then $\alpha \in \mathbb{F}_{3^n}$ is fixed and we want to have
\[ 0 = \text{Tr}_m(\alpha z) = \alpha z + \ldots + (\alpha z)^{3^m-1} + (\alpha z)^{3^m-1} + \ldots + (\alpha z)^{3^m-1} \]
\[ = (\alpha + \alpha^{3^m}) z + (\alpha^{3^m} + \alpha^{3^m+1}) z^3 + \ldots + (\alpha^{3^m-1} + \alpha^{3^m-1}) z^{3^m-1} \]
\[ = \text{Tr}_m((\alpha + \alpha^{3^m}) z) \]
On the normality of \( p \)-ary bent functions

for all \( z \in \mathbb{F}_{3^m} \) which is of the form \( z = \omega^{\ell (3^m+1) \frac{3^k+1}{2}} \), \( 0 \leq \ell \leq 3^m - 2 \). If we can find \( j \) such that \( \alpha + \alpha^{3^m} = 0 \), then \( f_{\ell}(x) = \text{Tr}_n(\omega^\ell x^{\frac{3^k+1}{2}}) = 0 \) on \( \omega^j \mathbb{F}_{3^m} \). Since \( \omega^j \mathbb{F}_{3^m} \) is a subspace of dimension \( m \), this implies then the normality of \( f_{\ell}(x) \). We have

\[
\alpha + \alpha^{3^m} = 0 \iff \omega^{3^m(\epsilon + \frac{3^k+1}{2}j)} = -\omega^{\frac{3^k+1}{2}j} \\
\iff \omega^{3^m(\epsilon + \frac{3^k+1}{2}j)} = \omega^{\epsilon + \frac{3^k+1}{2}j + \frac{3^{2m} - 1}{2}} \\
\iff \omega^{(3^m-1)(\epsilon + \frac{3^k+1}{2}j) - \frac{3^{2m} - 1}{2}} = 1 \\
\iff 3^{2m} - 1(3^m - 1)(\epsilon + \frac{3^k+1}{2}j) = \frac{3^{2m} - 1}{2} \\
\iff 3^m + 1|\epsilon + \frac{3^k+1}{2}j - \frac{3^m + 1}{2} \\
\iff \frac{3^k+1}{2}j \equiv \frac{3^m + 1}{2} - \epsilon \mod 3^m + 1.
\]

Since gcd\( (3^m + 1, \frac{3^k+1}{2}) = 2 \) and \( \frac{3^m + 1}{2} - \epsilon \) is even, this congruence always has exactly two solutions. Hence, there are at least two subspaces of dimension \( n/2 = m \) on which the Coulter-Matthews regular bent functions take the value 0.

\[ \square \]

3 Conclusion

We analyse normality of \( p \)-ary bent functions, show that normal bent functions must be regular or not weakly regular. We give a characterization of the quadratic monomials which belong to the completed Maiorana-McFarland class and hence obtain a generalization of Theorem 4 in [1]. We show that the regular Coulter-Matthews bent functions, which do not belong to the completed Maiorana-McFarland class, are though normal.

References

Abstract We generalize to the Arithmetic Walsh Transform (AWT) some results which were previously known for the Walsh Hadamard transform of Boolean functions. We firstly generalize the classical Poisson summation formula to the AWT. We secondly show that the AWT of a large class of Boolean functions can be expressed in terms of the AWT of a Boolean function of algebraic degree at most 3 in a larger number of variables.

Keywords Arithmetic Walsh Transform; Boolean function; Poisson summation formula

1 Introduction

The (standard) imbalance of a Boolean function is the number of times it takes the value 0 minus the number of times it takes the value 1. The Walsh-Hadamard transform (WHT) of a Boolean function \( f \) is the set of imbalances of functions of the form \( f + l \) where \( l \) is linear and the addition is in \( \mathbb{F}_2 \). The WHT of a Boolean function has been a subject of study for several decades. Much of the recent interest stems from its role in cryptography, where it serves as a measure of resistance to certain cryptanalytic attacks. The WHT also gives

This material is based upon work supported by the National Science Foundation under Grant No. CCF-0514660. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Claude Carlet
LAGA, Department of Mathematics, University of Paris 8 (and Paris 13 and CNRS),
2 rue de la liberté, 93526 Saint–Denis cedex 02, France.
E-mail: claude.carlet@univ-paris8.fr

Andrew Klapper
Dept. of Computer Science, 307 Marksbury Building, University of Kentucky, Lexington,
KY, 40506-0633. www.cs.uky.edu/
E-mail: klapper@cs.uky.edu
a characterization of bent functions (all the Walsh-Hadamard coefficients are $\pm 2^{n/2}$, where $n$ is the number of variables). It is desirable to have an understanding of what values can be taken on by the WHT of Boolean functions with particular properties (and not much is known with this respect).

Recently Mark Goresky and the second author defined an arithmetic or “with carry” analog of the WHT, known as the arithmetic Walsh transform (AWT) [4], described in Section 2. Its definition is related to the arithmetic correlation of sequences, which plays an important role for the sequence generators called FCSRs [3, page 69] and for arithmetic codes [7,8]. Many basic facts were worked out in [4]. In a later paper, the second author obtained the AWT of quadratic functions [5]. This paper continues the study of AWTs.

In earlier work, the first author showed that the imbalance of any Boolean function (and hence any Walsh-Hadamard coefficient) can be expressed in terms of the imbalance of a related Boolean function of degree at most 3, but depending on additional variables [1]. This result showed that the weights (and WHT values) of cubic functions (that is, of functions of algebraic degree 3) are much more complex than those of quadratic functions (of algebraic degree 2, see [6]). It was also used to construct a family of bent functions of degree 4 from a bent function of degree 3.

In this paper we present two results that further our understanding of the AWT. Although not explicitly connected to each other, both are analogs of well known results on the WHT. First we generalize the well-known Poisson summation formula from the WHT [2,6] to the AWT. This formula allowed the proof of many important properties of Boolean functions (such as deducing bounds on the algebraic degree of a Boolean function from divisibility properties of its WHT, characterizing correlation immune functions by their WHT, relating bent functions and their duals, studying propagation criteria [2]). We hope our generalization will allow similar properties in the framework of addition with carry to be proved in the future; such properties may be more difficult to guess (and finding them is probably a long term work) since the formula for the AWT is more complex than for the WHT and has different expressions depending on the relations between certain parameters. In the second part of the paper, we show that the AWT of a function $f$ can be expressed in terms of AWTs of functions with low degrees (similarly to the case of WHT, observed in [1]), under some restrictive condition on $f$.

2 Basics of Arithmetic Walsh Transforms

Arithmetic Walsh transforms (AWTs), a with-carry analog of WHTs, were first considered by Goresky and Klapper [4]. Let $\langle a, b \rangle$ denote the inner product modulo 2 of $a, b \in \{0, 1\}^n$. Let $[x]_2$ denote the reduction modulo 2 of $x$, where $x$ is an integer\(^1\). Let $B_n$ be the set of Boolean functions on $V_n = \mathbb{F}_2^n = \{0, 1\}^n$.

\(^1\) Since we sometimes treat Boolean values as integers, it is helpful to indicate when we want to reduce an expression involving such values modulo 2.
Recall that $R_n$ is the ring of Boolean valued functions $f$ on $\mathbb{N}^n$, the set of $n$-tuples of nonnegative integers, with an algebraic structure based on arithmetic with carry. In this algebra, to add $f_1$ and $f_2$, we add the values of $f_1$ and $f_2$ at $c \in \mathbb{N}^n$ modulo 2, but we retain a “carry” to the value of $f_1 + f_2$ at $c + 1^n$. It was shown that an element $f$ of $R_n$ corresponds to a choice of a 2-adic integer

$$\bar{f}(c) = \sum_{i=0}^{\infty} f(c + i \cdot 1^n) 2^i$$

for every $c \in X_n = \{c \in \mathbb{N}^n : c_i = 0$ for some $i\}$. That is, we can gather together all the values on the diagonal $D_c = \{c + i \cdot 1^n : i = 0, 1, \ldots\}$. Addition in $R_n$ then corresponds to 2-adic addition of corresponding 2-adic diagonal elements. Note that the addition and subtraction of 2-adic numbers are not the same operation. If $a = \sum_{i=0}^{\infty} a_i 2^i$ and $b = \sum_{i=0}^{\infty} b_i 2^i$, then $c = a - b = \sum_{i=0}^{\infty} c_i 2^i$ can be found iteratively as follows

- $c_0 \in \{0, 1\}$ is defined by $a_0 + 2d_0 = b_0 + c_0$, where $d_0 \in \{0, 1\}$;
- $c_1 \in \{0, 1\}$ is defined by $a_1 + 2d_1 = b_1 + c_1 + d_0$, where $d_1 \in \{0, 1\}$;
- We continue infinitely in this way.

It is well known that the sum and difference of $k$-periodic $N$-adic numbers are eventually $k$-periodic. Since this plays an important role in the paper, we recall the idea of the proof, for differences. Two $N$-adic numbers $a$ and $b$ are $k$-periodic if $a = -u/(N^k - 1)$ and $b = -v/(N^k - 1)$ with $u, v \in \{0, 1, ..., N^k - 1\}$. Then $a - b = (v - u)/(N^k - 1)$. The result then follows from Theorem 4.2.4 of Goresky and Klapper’s book [3, p. 75]: we know that $-(N^k - 1) \leq v - u \leq N^k - 1$; if $v - u \leq 0$, then $(v - u)/(N^k - 1)$ is $k$-periodic; otherwise it can be written as $1 + w/(N^k - 1)$ with $-(N^k - 1) < w \leq 0$. The $N$-adic expansion of $w/(N^k - 1)$ for such $w$ is $k$-periodic, and is not the all $N - 1$ sequence (which would give $w = -(N^k - 1)$). Thus in adding 1 to $w/(N^k - 1)$ the carries cannot propagate past the first non-$(N - 1)$ coefficient, hence cannot propagate beyond the first $k$ coefficients. Beyond that the coefficients of $1 + w/(N^k - 1)$ are the same as the coefficients of $w/(N^k - 1)$, hence are periodic.

An element $f \in R_n$ is 2-periodic if $f(c + 2d) = f(c)$ for every $c, d \in \mathbb{N}^n$. It is eventually 2-periodic if $f(c + 2d) = f(c)$ for all but finitely many $c$, for all $d$. If $f$ is eventually 2-periodic, then a complete period of $f$ is any set $S$ of $2^n$ vectors $c$ such that

1. $\forall d \in \mathbb{N}^n$, $f(c + 2d) = f(c)$ (that is, $c$ is in the periodic part of $f$), and
2. $\forall d \in \mathbb{N}^n$, there is a (necessarily unique) $c \in S$ so that $c \equiv d \mod 2$.

Let $U_n = \{c \in \{0, 1\}^n : c_1 = 0\}$. If $f$ is 2-periodic, then it is determined by the values $\bar{f}(c)$ with $c \in U_n$. For example, if $n = 2$ then $f(1, 0) = f(1, 2)$ (by 2-periodicity) = $f(0, 1) + 1^n)$ so $f(1, 0)$ is determined by a value on the diagonal starting at $(0, 1)$. In general if $f$ is 2-periodic, then

$$\bar{f}(c) = f(c) + f(c + 1^n)2 + f(c)2^2 + f(c + 1^n)2^3 + \cdots$$

$$= (f(c) + 2f(c + 1^n))(1 + 4 + 4^2 + \cdots)$$

$$= -\frac{f(c) + 2f(c + 1^n)}{3}$$
shifted arithmetic autocorrelations of an \( \ell \) of the sequences and gives nice properties, as for instance the fact that the periodic part also plays a role but this definition captures the main behavior is taken into account as well. This may seem rather restrictive since the non-correlation of sequences [3, page 178], [7,8] in which only the periodic part 2-adic integer, and this is determined by the periodic part.

integer, which equals the denominator of the rational representation of the size of and circuit complexity of an FCSR is determined by the connection integer, which equals the denominator of the rational representation of the 2-adic integer, and this is determined by the periodic part.

Let \( f \in R_n \) be a nontrivial linear subspace of \( V_n = \mathbb{F}_2^n = \{0,1\}^n \) (which is an \( \mathbb{F}_2 \)-vector space under addition mod 2). Let \( S^\perp \) denote the set of vectors that are orthogonal to every vector in \( S \) (the dual of code \( S \)). Recall that \( \dim(S^\perp) = n - \dim(S) \). For any \( d \subseteq V_n \), let \( \langle d \rangle = \{0^n, d\} \) denote the linear subspace generated by \( d \). If \( d \in V_n \) and \( S \) and \( T \) are linear subspaces of \( V_n \) then \( d + S = \{d + a : a \in S\} \) and \( S + T = \{a + b : a \in S, b \in T\} \).

As recalled in the introduction, this definition is related to the arithmetic correlation of sequences [3, page 178], [7,8] in which only the periodic part is taken into account as well. This may seem rather restrictive since the non-periodic part also plays a role but this definition captures the main behavior of the sequences and gives nice properties, as for instance the fact that the shifted arithmetic autocorrelations of an \( \ell \)-sequence are identically zero. Also, the size of and circuit complexity of an FCSR is determined by the connection integer, which equals the denominator of the rational representation of the 2-adic integer, and this is determined by the periodic part.

Let \( f \in B_n \). We extend \( f \) to a 2-periodic function \( \bar{f} \in R_n \) by letting \( \bar{f}(a) = f(a \mod 2) \). We also let \( \bar{f}_c(a) = \langle a, c \rangle \) be a linear function, whose extension to \( \mathbb{N}^n \) is denoted \( \bar{f}_c \), and whose associated 2-adic number on the diagonal based at \( b \) is \( \bar{l}_a(b) \). Note that \( \bar{f} - \bar{f}_c \in R_n \) is eventually 2-periodic. Then the arithmetic Walsh coefficient of \( f \) at \( c \) is

\[
W(f)(c) = Z(f - \bar{f}_c) = \sum_{b \in U_n} z(\bar{f}(b) - \bar{l}_a(b)) = \frac{1}{2} \sum_{b \in V_n} z(\bar{f}(b) - \bar{l}_a(b)),
\]

and the arithmetic Walsh transform of \( f \) is \( W(f) = \{W(f)(c) : c \in \{0,1\}^n\} \).

3 Poisson Summation Formula

Let \( S \) be a nontrivial linear subspace of \( V_n = \mathbb{F}_2^n = \{0,1\}^n \) (which is an \( \mathbb{F}_2 \)-vector space under addition mod 2). Let \( S^\perp \) denote the set of vectors that are orthogonal to every vector in \( S \) (the dual of code \( S \)). Recall that \( \dim(S^\perp) = n - \dim(S) \). For any \( d \subseteq V_n \), let \( \langle d \rangle = \{0^n, d\} \) denote the linear subspace generated by \( d \). If \( d \in V_n \) and \( S \) and \( T \) are linear subspaces of \( V_n \) then \( d + S = \{d + a : a \in S\} \) and \( S + T = \{a + b : a \in S, b \in T\} \).
Let \( \tilde{f}(a) = \sum_{x \in V_n} (-1)^{f(x) + a \cdot x} \) be the Walsh-Hadamard transform of \( f \) at input \( a \in V_n \). In classical theory of Boolean functions [2,6], the Poisson summation formula says that if \( f \in B_n \) is a Boolean valued function on \( V_n \), then for any \( d \in V_n \),

\[
\sum_{a \in S} (-1)^{d \cdot a} \tilde{f}(a) = 2^{\dim(S)} \sum_{b \in d + S^\perp} (-1)^f(b) = 2^n - 2^{|S|} \sum_{b \in d + S^\perp} f(b).
\]

In this section, we consider similar summation formulas for the arithmetic Walsh transform \( W(f)(a) \). For a fixed \( d \in V_n \), let

\[
\Gamma_S(f) = \sum_{a \in S} (-1)^{d \cdot a} W(f)(a) = \frac{1}{2} \sum_{a \in S} (-1)^{d \cdot a} \sum_{b \in V_n} z(\tilde{f}(b) - \tilde{I}_a(b)).
\]

We have

\[
\Gamma_S(f) = \frac{1}{2} \sum_{b \in V_n} \sum_{a \in S} (-1)^{d \cdot a} \delta_{a,b},
\]

where \( \delta_{a,b} = z(\tilde{f}(b) - \tilde{I}_a(b)) = z\left(-\frac{f(b) + 2f(b + 1^n)}{3} + \frac{I_a(b) + 2I_a(b + 1^n)}{3}\right) \). Denoting \( \psi_{b,x} = z(\tilde{f}(b) + x) \), we have four cases:

1. If \( a \cdot 1^n = 0 \) and \( a \cdot b = 0 \), then \( \delta_{a,b} = \psi_{b,0} \).
2. If \( a \cdot 1^n = 0 \) and \( a \cdot b = 1 \), then \( \delta_{a,b} = \psi_{b,1} \).
3. If \( a \cdot 1^n = 1 \) and \( a \cdot b = 0 \), then \( \delta_{a,b} = \psi_{b,2/3} \).
4. If \( a \cdot 1^n = 1 \) and \( a \cdot b = 1 \), then \( \delta_{a,b} = \psi_{b,1/3} \).

In order to calculate these expressions, we first need to determine the values of \( z(w/3) \) when \( w = -3, \ldots, 3 \). They are given below. For instance, \( z(-3/3) \) corresponds to \( f(c) = f(c + 1^n) = 1 \) in (1) and \( z(1/3) = z(1 - 2/3) = z(-2/3) \).

<table>
<thead>
<tr>
<th>( w )</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z(w/3) )</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

This allows the value of \( \psi_{b,x} \) to be calculated in each of the four cases \( x = 0, 1, 2/3, 1/3 \) and for each pair \( (f(b), f(b + 1^n)) \).

| \( f(b) \) | 0 | 0 | 1 | 1 |
|---|---|---|---|
| \( f(b + 1^n) \) | 0 | 1 | 0 | 1 |
| \( \psi_{b,0} \) | 2 | 0 | 0 | -2 |
| \( \psi_{b,1} \) | 2 | 0 | 0 | 2 |
| \( \psi_{b,2/3} \) | 0 | 2 | 0 | 0 |
| \( \psi_{b,1/3} \) | 0 | 0 | 2 | 0 |

We are then able to give the expression of \( \psi_{b,x} \) in each case, by means of \( (f(b), f(b + 1^n)) \). We have \( \psi_{b,0} = (-1)^f(b) + (-1)^f(b + 1^n) = 2(1 - f(b) - f(b + 1^n)) \).
The expressions for $\psi_{b,x}$ in the three other cases are obtained by doing Lagrange interpolation: treat $f(b)$ and $f(b + 1^n)$ as variables $u, v$; then (for example) to find $\psi_{b,1}$ we want a polynomial $h(u, v)$ so that $h(0,0) = h(1,1) = 2, h(0,1) = h(1,0) = 0$. Using Lagrange interpolation, one finds $h(u, v) = 2uv + 2(1 - u)(1 - v) = 2(1 - u - v + 2uv)$. Thus $\psi_{b,1} = 2(1 - f(b) - f(b + 1^n) + 2f(b)f(b + 1^n))$. Doing similarly in the other cases, we have

$$\psi_{b,x} = \begin{cases} 
2(1 - f(b) - f(b + 1^n)) & \text{if } x = 0, \\
2(1 - f(b) - f(b + 1^n) + 2f(b)f(b + 1^n)) & \text{if } x = 1, \\
2f(b)(1 - f(b + 1^n)) & \text{if } x = 2/3, \\
2f(b)(1 - f(b + 1^n)) & \text{if } x = 1/3.
\end{cases}$$

**Notation:** For any set $T \subseteq V_n$, let $H_T = \sum_{b \in T} f(b)$ and $Q_T = \sum_{b \in T} f(b)f(b + 1^n)$.

3.1 In the case where $a \cdot 1^n = 0$ for every $a \in S$, we have:

$$\Gamma_S(f) = \frac{1}{2} \sum_{b \in V_n} \sum_{a \in S, a \cdot b = 0} (-1)^{d \cdot a} \psi_{b,0} + \frac{1}{2} \sum_{b \in V_n} \sum_{a \in S, a \cdot b = 1} (-1)^{d \cdot a} \psi_{b,1}.$$  

Let $T_{b,d} = \sum_{a \in S, a \cdot b = 1} (-1)^{d \cdot a}$. Then

$$\Gamma_S(f) = \sum_{b \in V_n} (1 - f(b) - f(b + 1^n)) \sum_{a \in S, a \cdot b = 0} (-1)^{d \cdot a} + (1 - f(b) - f(b + 1^n) + 2f(b)f(b + 1^n)) \sum_{a \in S, a \cdot b = 1} (-1)^{d \cdot a}$$  

$$= \sum_{b \in V_n} (1 - f(b) - f(b + 1^n)) \sum_{a \in S} (-1)^{d \cdot a} + 2f(b)f(b + 1^n)T_{b,d}$$  

$$= (2^n - 2H_{V_n}) \sum_{a \in S} (-1)^{d \cdot a} + 2 \sum_{b \in V_n} f(b)f(b + 1^n)T_{b,d}.$$  

**Lemma 1** If $d \in S^\perp$, then $T_{b,d} = 0$ if $b \in S^\perp$ and $T_{b,d} = |S|/2$ if $b \not\in S^\perp$. If $d \not\in S^\perp$, then $T_{b,d} = 0$ if $b \in S^\perp$ or $b \in V_n \setminus (\langle d \rangle + S^\perp)$ and $T_{b,d} = -|S|/2$ if $b \in (\langle d \rangle + S^\perp) \setminus S^\perp = d + S^\perp$. Also, if $d \in S^\perp$, then $\sum_{a \in S} (-1)^{d \cdot a} = |S|$, otherwise $\sum_{a \in S} (-1)^{d \cdot a} = 0$.

**Proof.** If $b \in S^\perp$, then the sum is empty, hence zero. Suppose $b \not\in S^\perp$. If $d \in S^\perp$, then $T_{b,d} = |\{a \in S : a \cdot b = 1\}| = |S|/2$. If $d \not\in S^\perp$ and $b \in V_n \setminus (\langle d \rangle + S^\perp)$, then $d$ is an independent parity check on $\{a \in S : a \cdot b = 1\}$, so $T_{b,d} = 0$. If $d \not\in S^\perp$ and $b = d + c$ with $c \in S^\perp$, then all terms in the sum are $-1$, so $T_{b,d} = -|S|/2$. The proof of the last statement is well-known and similar. \[\Box\]

We directly deduce the following theorem.

**Theorem 1** Suppose that $1^n$ is a parity check for $S$. Then

$$\Gamma_S(f) = \begin{cases} |S|(2^n - 2H_{V_n} + Q_{V_n \setminus S^\perp}) & \text{if } d \in S^\perp, \\
-|S|Q_{d + S^\perp} & \text{if } d \not\in S^\perp.
\end{cases}$$

This seems simple enough to hope to deduce nice properties in the future.
Theorem 2 For all $d \in V_n$, we have $T_{b,d,0} = 0$ if $b \in S_0^d$ and $T_{b,d,1} = 0$ if $b \in S^d$. If $d \in S^d$, then $T_{b,d,0} = |S|/4$ if $b \notin S^d$ and $T_{b,d,1} = \{\frac{|S|}{2} \text{ if } b \in S_0^d \setminus S^d, 0 \text{ else}\}$. If $d \in S_0^d \setminus S^d$, then $T_{b,d,0} = 0$ if $b \notin S^d$ and $T_{b,d,1} = -|S|/2$ if $b \in S_0^d \setminus S^d$ and is 0 if $b \notin S_0^d$. If $d \notin S_0^d$, then $T_{b,d,0} = T_{b,d,1} = 0$ if $b \notin (d) + S_0^d$, $T_{b,d,1} = 0$ if $b \in S_0^d \setminus S^d$, $T_{b,d,0} = -|S|/4$ if $b \in d + S_0^d$, $T_{b,d,1} = -|S|/4$ if $b \in d + S^d$, and $T_{b,d,1} = |S|/4$ if $b \in d + (S_0^d \setminus S^d)$. 

Proof. In the first assertion the sums are empty, hence 0. Let $A_1 = |\{a \in S_i : a \cdot b = 1\}|$ and $B_1 = |\{a \in S_i : a \cdot b = a \cdot d = 0\}|$ so $T_{b,d,i} = A_1 - 2B_1$. 

Suppose $d \in S^d$. Then $T_{b,d,0} = A_1$. If $b \notin S_0^d$, then $a \cdot b = 1$ for half the $a \in S_i$, so $A_1 = \frac{|S_i|}{2} = \frac{|S|}{4}$. Hence, $T_{b,d,0} = T_{b,d,1} = \frac{|S|}{4}$. If $b \in S_0^d \setminus S^d$, then $a \cdot b = 1$ if $a \in S_1$. Hence, $T_{b,d,1} = |S_1| = \frac{|S|}{2}$. 

Suppose $d \in S_0^d \setminus S^d$. If $b \notin (d) + S_0^d$, then $A_0 = |S_0^d|/2 = \frac{|S|}{4}, A_1 = |S_1|/2 = \frac{|S|}{4}, B_0 = A_0/2 = \frac{|S|}{8}$ and $B_1 = A_1/2 = \frac{|S|}{8}$, so $T_{b,d,i} = 0, i = 0, 1$. If $b \in S_0^d \setminus S^d$, then $A_1 = |S_1| = \frac{|S|}{2}$ and $B_1 = |S_1|/2 = \frac{|S|}{4}$, so $T_{b,d,1} = 0$. If $b \in d + S^d$, then $A_0 = A_1 = B_0 = B_1 = \frac{|S|}{2} = \frac{|S|}{4}$, so $T_{b,d,0} = T_{b,d,1} = -\frac{|S|}{4}$. If $b \in d + (S_0^d \setminus S^d)$, then $A_0 = A_1 = B_0 = B_1 = \frac{|S|}{2} = \frac{|S|}{4}$ and $B_1 = 0$, so $T_{b,d,0} = -\frac{|S|}{4}$ and $T_{b,d,1} = \frac{|S|}{4}$. 

We can deduce the following theorem.
Theorem 2 Suppose that $1^n$ is not a parity check for $S$. Then, with the notation above:

If $d \in S^\perp$: $\Gamma_S(f) = (|S|/4)(2^{n+1} - 2H_{V_n} - 2Q_{S_0^\perp} + H_{S_0^\perp} - 2H_{S^\perp} - H_{1^n + S^\perp} + 2H_{1^n + d + S_0^\perp})$.

If $d \in S_0^\perp \setminus S^\perp$: $\Gamma_S(f) = (|S|/2)(2^{n} - 3H_{V_n} + Q_{V_n} + H_{1^n + S_0^\perp} - H_{1^n + S^\perp} - H_{S_0^\perp} + H_{S^\perp})$.

If $d \notin S_0^\perp$: $\Gamma_S(f) = -(|S|/4)(2Q_{d + S_0^\perp} + 2H_{d + S^\perp} - 2H_{1^n + d + S_0^\perp} + H_{1^n + d + S_0^\perp})$.

4 AWT and Cubic Boolean Functions

In this section we show that, under some conditions, the arithmetic Walsh coefficient can be realized as the 2-adic imbalance of a Boolean function of degree at most three, but possibly in more variables.

Lemma 3 Let $f \in B_n$ be a Boolean function and let $f'(a, b, c) = f(a) + bc \in B_{n+2}$, where $b$ and $c$ are additional Boolean variables. Then $Z(f') = 2Z(f)$.

Proof. Considering the cases $(b, c) = (0, 0), (0, 1), (1, 0)$ and $(1, 1)$, we have that

$$Z(f') = \sum_{a \in \mathbb{F}_n} \sum_{b, c \in \{0, 1\}} (-1)^{f(a) + bc} + (-1)^{f(a + 1^n) + (b + 1)(c + 1)}$$

$$= \sum_{a \in \mathbb{F}_n} \left[ (-1)^{f(a)} - (-1)^{f(a + 1^n)} + (-1)^{f(a)} + (-1)^{f(a + 1^n)} + (-1)^{f(a)} ight] + (-1)^{f(a + 1^n)} - (-1)^{f(a)} + (-1)^{f(a + 1^n)} \right] = 2Z(f).$$

Assume now that $f(a) = f_1(a)f_2(a) + g(a)$ and apply a translation $b \mapsto b + f_2(a)$ and $c \mapsto c + f_1(a)$. We get

$$Z(f') = \frac{1}{2} \sum_{b, c \in \mathbb{F}_2} \sum_{a \in \{0, 1\}^n} z(f'(a, b, c))$$

$$= \frac{1}{2} \sum_{b, c \in \mathbb{F}_2} \sum_{a \in \{0, 1\}^n} \sum_{b, c \in \mathbb{F}_2} A_{a, b, c}$$

with $A_{a, b, c} = z \left( -\frac{[f(a) + (b + f_2(a))(c + f_1(a))]}{3} + 2[f(a + 1^n) + (b + f_2(a) + 1)(c + f_1(a) + 1)] \right)$.

This expression is the 2-adic imbalance of a Boolean function when $f_i(a + 1^n) = f_i(a)$ for all $a$ and for $i = 1, 2$.

Definition 1 We say that $f \in B_n$ is diagonal if it has $1^n$ as a linear structure. That is, if for all $a \in \{0, 1\}^n$ we have $f(a + 1^n) = f(a)$. Similarly, $f \in R_n$ is diagonal if for all $a \in \mathbb{N}^n$ we have $f(a + 1^n) = f(a)$. We say $f \in R_n$ is eventually diagonal if for all but finitely many $a \in \mathbb{N}^n$ we have $f(a + 1^n) = f(a)$. 
For any function $f$, function $\Delta f(a) = f(a) + f(a + 1^n)$ is diagonal. Every diagonal function $g$ is $\Delta f$ for some $f$, and $\Delta f = 0$ if and only if $f$ is diagonal.

**Theorem 3** Suppose $f$ is a diagonal Boolean function with degree $k$. Then there are an integer $p$ and diagonal Boolean functions $f_{i,j}$, $i = 1, 2$, $j = 1, \cdots , p$, such that $f = \sum_{j=1}^{p} f_{i,j} f_{2,j}$ and $\deg(f_{i,j}) \leq \lceil k/2 \rceil$.

**Proof.** It suffices to show this for $f = \Delta c_1 \cdots c_{k+1}$, where $c = (c_1, \cdots , c_n)$. We prove this by induction on $k$. If $k = 0$, then $f = 1$ so $f_{1,1} = f_{2,1} = 1$.

Suppose the result is true for $\Delta c_1 \cdots c_m$, $m < k$. We claim that

$$\Delta c_1 \cdots c_{k+1} = \Delta c_1 \cdots c_k + \prod_{i=1}^{k} (c_i + c_{k+1}).$$

Indeed, the left hand side includes all monomials in $c_1, \cdots , c_{k+1}$ of degrees at most $k$. The first term on the right hand side includes all monomials in $c_1, \cdots , c_k$ of degree less than $k$, and the second term on the right hand side includes all monomials in $c_1, \cdots , c_{k+1}$ that are multiples of $c_{k+1}$ and have degrees at most $k$, as well as the monomial $c_1 \cdots c_k$.

By induction there are an integer $p$ and diagonal Boolean functions $f_{i,j}$, $i = 1, 2$, $j = 1, \cdots , p$, such that $\Delta c_1 \cdots c_k = \sum_{j=1}^{p} f_{i,j} f_{2,j}$ and $\deg(f_{i,j}) \leq \lceil (k-1)/2 \rceil$. We can write

$$f_{1,p+1} = \prod_{i=1}^{\lceil k/2 \rceil} (c_i + c_{k+1}) \quad \text{and} \quad f_{2,p+1} = \prod_{i=\lceil k/2 \rceil+1}^{k} (c_i + c_{k+1}).$$

Then

$$\Delta c_1 \cdots c_{k+1} = \sum_{j=1}^{p+1} f_{i,j} f_{2,j},$$

proving the theorem. \qed

**Corollary 1** Let $g \in R_n$ be eventually 2-periodic and eventually diagonal. Then there are an integer $p \geq 1$ and a diagonal Boolean function $h \in B_{n+2p}$ such that $Z(h) = 2^p Z(g)$ and $\deg(h) \leq 3$.

**Proof.** Since $g \in R_n$ is eventually 2-periodic, there is a unique Boolean function $f$ so that $f(a) = g(a)$ for all but finitely many $a \in \{0,1\}^n$. Moreover $f$ is diagonal and $Z(f) = Z(g)$.

Let $k$ be the degree of $f$. If $k \leq 3$, then we are done. Otherwise we apply Theorem 3 to obtain diagonal Boolean functions $f_{i,j}$, $i = 1, 2$, $j = 1, \cdots , p$, such that $f = \sum_{j=1}^{p} f_{i,j} f_{2,j}$ and $\deg(f_{i,j}) \leq \lceil k/2 \rceil$.

Let $\langle x, y \rangle$ denote the inner product of $x$ and $y$. Let $b = (b_1, \cdots , b_p)$, $c = (c_1, \cdots , c_p)$, and $f_i(b) = (f_{i,1}(b_1), \cdots , f_{i,p}(b_p))$, $i = 1, 2$. Let

$$f'(a, b, c) = [f(a) + \langle b, c \rangle]_2 \in B_{n+2p}, \quad \text{and} \quad f''(a, b, c) =$$
\[ [f(a) + (b + f_2(a), c + f_1(a))]_2 = \sum_{j=1}^{p} (b_j c_j + b_j f_{1,j}(a) + c_j f_{2,j}(a)) \mod 2. \]

By Lemma 3 we have \( Z(f') = 2^p Z(f) \). As in equation (2) but with \( b, c \in \{0, 1\}^p \) instead of in \( \{0, 1\} \), we have

\[
Z(f') = \frac{1}{2} \sum_{a \in \{0, 1\}^n} \sum_{b, c \in \{0, 1\}^p} z \left( \frac{[f(a) + (b + f_2(a), c + f_1(a))]_2}{3} - \frac{2([f(a + 1^n) + (b + 1^p + f_2(a), c + 1^p + f_1(a))]_2}{3} \right).
\]

We have \( [f(a) + (b + f_2(a), c + f_1(a))]_2 = f''(a, b, c) \) and

\[
[f(a + 1^n) + (b + 1^p + f_2(a), c + 1^p + f_1(a))]_2 = [f''(a + 1^n, b + 1^p, c + 1^p)]_2.
\]

Thus \( Z(f'') = Z(f') = 2^p Z(f) \), and \( \deg(f'') \leq 1 + [k/2] \).

We can repeat these steps, replacing \( f \) by \( f'' \), until the degree is at most 3. \( \square \)

**Corollary 2** Let \( f \in B_n \) be a diagonal Boolean function and let \( c \in V_n \). Then there are an integer \( p \geq 0 \) and a Boolean function \( h \in B_{n+2p} \) so that \( h \) has algebraic degree at most 3 and \( Z(h) = 2^p W(f)(c) \).

**Proof.** Apply Corollary 1 to \( g = f - 1_c \). \( \square \)

In [5] has been determined the AWT of quadratic functions which is peculiar. Corollary 2 shows that the AWT of cubic functions behaves more like that of general functions, as it was already the case with the WHT.

**References**

Fast Multiplication of the Algebraic Normal Forms of Two Boolean Functions.

Subhabrata Samajder · Palash Sarkar

Abstract The contribution of this paper is twofold. Firstly, it proposes a simple algorithm which performs the multiplication of two $n$-variate boolean functions in their algebraic normal forms in $O(2^n)$ time and $O(2^n)$ space. Secondly, it proposes a fast implementation (MultANF$_w$) of the algorithm which works with $w$-bit words. Results for $w = 8, 32$ and $64$ show that the 64-bit implementation is the fastest. To further analyze the performance, a sparse implementation has been done, which we call quadratic implementation.

It has been observed that for a $w$-bit implementation, if the product of the number of monomials of the two input polynomials is $< 2^{n - \log_2 w}$, then the quadratic implementation performs better than MultANF$_w$. It is also found that MultANF$_w$ performs much better than the algorithm internally used by Sage for all the three variants, i.e., $w = 8, 32$ and $64$. Our study also indicates that quadratic implementation performs better than SAGE.

Keywords Multivariate Polynomial Multiplication · Boolean Functions · Algebraic Normal Form (ANF)

1 Introduction

Let $R = GF(2)[x_1, x_2, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$. We consider polynomials in $R$. Such polynomials can be considered to be the algebraic normal form of $n$-variable Boolean functions, which are maps from $\{0, 1\}^n$ to $\{0, 1\}$. Multiplication of Boolean functions is a basic operation and is of interest in itself. Apart from this, it also has a wide range of applications.
The Buchberger’s algorithm ([Buc06], [Buc98]) and its improvements, the $F_4$ and $F_5$ algorithms ([Fau99], [Fau02]), to compute the Gröbner basis over $\mathbb{R}$ essentially use polynomial multiplications to cancel out the leading terms. Hence, improving upon polynomial multiplications over $\mathbb{R}$, will speed up these algorithms.

The algebraic immunity ([CM03], [MPC04], [Dal06]) of Boolean functions, is crucial to the security of the block ciphers and stream ciphers. The definition of algebraic immunity says that the algebraic immunity for a Boolean function $f$ is the minimum degree of $g$, such that $f.g = 0$. Thus, one can see that improving the time taken to multiply two Boolean functions in their ANF’s has direct application to algebraic immunity. It also has applications in non-linear codes, like higher order Reed-Muller Codes and Kerdock Codes ([PMS+98]).

Multiplication of two sparse polynomials $p$ and $q$ having $l_p$ and $l_q$ terms each will have about $l_p l_q$ terms and so the usual algorithm which takes $O(l_p l_q)$ time, is optimal. It would be nice to investigate whether this can be improved in case of dense polynomials, where the number of variables is, say 30.

Our Results: A simple observation leads to an $O(n 2^n)$ time and $O(2^n)$ space recursive algorithm. Asymptotically, this is competitive with general purpose Fourier transform based multivariate polynomial multiplication algorithm ([Mat08]) specialized to the binary case. To the best of our knowledge, the binary case does not seem to have received separate attention. On the other hand, for cryptographic application, the binary case is arguably the most important case.

Our contribution is not only in identifying a simple algorithm for multiplication of ANF’s of Boolean functions, but also, in carrying out high quality software implementation. We make a careful study of the algorithm and identify ways to speed up. The first issue is to avoid recursion. For this we simulate the recursion tree independently for each of the two input polynomials $p$ and $q$. We call this as our pre-process step. Next, instead of bit level AND operations, 8-bit table lookups are used to multiply two 3-variate polynomials at once. After table lookups we again traverse up the recursion tree by doing similar set of operation to finally get the product $pq$. This step is called the post-process step.

Notice that the polynomials, can also be seen as a sequence of bits. To make use of the $w$-bit word arithmetic and hence improve speed, the polynomials are packed in $w$-bit words. Three different implementations of our algorithm is proposed, by taking $w = 8, 32$ and 64. A detailed comparison amongst these three implementations is given. Comparison with the software package SAGE shows that our implementations work much better than SAGE.

We have also done an efficient sparse implementation, which we call the quadratic implementation. It was then compared with the $w$-bit implementations mentioned above. We found that for the $w$-bit implementation if the number $l_p l_q$ is greater or equal to $2^{n - \log_2 w}$, then the MultANF$_w$ algorithm performs better than the quadratic implementation. For sparse case also, we
have compared the quadratic implementation with that of SAGE and found that the quadratic implementation works better than SAGE.

The organization is as follows: in Section 2, we give the basic idea and the propose ways to further improve upon our basic idea. A non-recursive \( w \)-bit implementation \( \text{MultANF}_w \) is proposed in Section 3. In Section 4, we give a detailed comparison of \( \text{MultANF}_w \), with its variants and with SAGE. Lastly, in Section 5, we conclude this paper.

2 The Algorithm

In the first half of this section we give the basic idea, then we describe an iterative algorithm for multiplying two boolean functions in their ANF’s and lastly, we conclude the section by pointing out ways in which we can further improve our iterative algorithm.

2.1 Basic Idea

Let, \( p(x_1, \ldots, x_n), q(x_1, \ldots, x_n) \in \mathbb{R} \). Write,

\[
p(x_1, \ldots, x_n) = x_n \cdot p_1(x_1, \ldots, x_{n-1}) \oplus p_0(x_1, \ldots, x_{n-1}) \\
q(x_1, \ldots, x_n) = x_n \cdot q_1(x_1, \ldots, x_{n-1}) \oplus q_0(x_1, \ldots, x_{n-1}).
\]

Then,

\[
pq = (p_1q_1) x_n^2 \oplus (p_1q_0 \oplus p_0q_1) x_n \oplus p_0q_0 \\
= (p_1q_1 \oplus p_1q_0 \oplus p_0q_1) x_n \oplus p_0q_0; \quad \text{[Since, } x_n^2 = x_n \text{ in } \mathbb{R}] \\
= \{(p_1 \oplus p_0)(q_1 \oplus q_0) \oplus p_0q_0\} x_n \oplus p_0q_0.
\]

Thus, the number of \((n - 1)\)-variate multiplications required is 2 instead of 4 at the cost of one extra addition.

\textbf{Note 1} This is a very simple observation and leads naturally to a fast recursive algorithm for multiplication of two ANF’s. To the best of our knowledge, it does not seem that the literature records this approach for multiplication of ANF’s.

Let, \( t(n) \) denote the time taken to multiply two \( n \)-variate polynomials and \( e(n) \) denote the time taken to add two \( n \)-variate polynomial. Then, we have

\[
t(n) = 2t(n - 1) + 4e(n - 1).
\]

Solving, we get

\[
t(n) = 2^n t(0) + 4 \times \{ e(n - 1) + 2 \times e(n - 2) + 2^2 \times e(n - 3) + \cdots \\
+ 2^{n-2} \times e(1) + 2^{n-1} \times e(0) \}.
\]
Since, $e(n) = 2^n \cdot e(0)$, using this we get,

$$t(n) = 2^n t(0) + 4n2^{n-1}e(0),$$

where, $t(0)$ and $e(0)$ denote the time taken for bit-wise AND and XOR. Therefore,

$$t(n) = O(n2^n) = O(2^{n+\log_2 n}) = O(m \log_2 m),$$

where $m = 2^n$. For “dense” polynomials, the size of the input will be about $O(m)$ and so this $O(m \log_2 m)$ algorithm is very attractive.

On the other hand, if $p$ and $q$ are “sparse” having $l_p$ and $l_q$ monomials respectively, then one would expect the product to have about $l_pl_q$ monomials. The direct algorithm for multiplication will require $O(l_pl_q)$ time and is about the best that one can expect. So the above $O(m \log_2 m)$ time algorithm is better only if the two polynomials are “dense”. More comparative details are given later.

2.2 An Iterative Algorithm

We represent polynomials in $\mathbb{R}$ using a sequence of bits. In this sequence, we denote the presence of every monomial by a single bit. Since the number of such possible monomials in $\mathbb{R}$ is $2^n$, we thus use $2^n$ bits to represent any polynomial in $\mathbb{R}$.

It is clear that one can compute the values of $p_0$, $(p_0 \oplus p_1)$, $q_0$ and $(q_0 \oplus q_1)$ independently and then multiply them to get the required $p_0q_0$ and $(p_0 \oplus p_1) \cdot (q_0 \oplus q_1)$. Thus one needs to compute $p_0$ and $p_0 \oplus p_1$ (respectively, $q_0$ and $q_0 \oplus q_1$) from $p$ (respectively, $q$). Using the same idea recursively, we thus get two recursive tree (one each for $p$ and $q$). Notice that both $p_0$ and $p_1 \oplus p_0$ are polynomials in $n - 1$ variables, namely $x_1, \ldots, x_{n-1}$. Thus, we see that at every step of the recursion the number of variables gets reduced by 1.

Suppose, $p$ is represented by a $2^n$ bit array $A$ (say). Then, $p_0$ corresponds to the first $2^{n-1}$ bits of $p$ and $p_1$ the last $2^{n-1}$ bits of $p$. Hence, $p_0 \oplus p_1$ is nothing but bit-wise XOR of the 1st half with the 2nd half of $A$. In the next step of the recursion tree, we repeat the same, each for $p_0$ and $p_0 \oplus p_1$ with $n$ now reduced to $n - 1$. We go on doing this until $n = 1$. In which case, we reach the leaves of the recursion tree. Notice that multiplication here is equivalent to bit-wise AND-ing. Thus to multiply two polynomials $p$ and $q$, we first build two recursion tree independently, each having $2^n$ 1-bit leaf nodes and then do a bit-wise AND among the corresponding leaf nodes of $p$ and $q$. Now, to get the final result $pq$, we traverse upwards from the leaves (which contains the bit-wise AND of the corresponding leaf nodes of $p$ and $q$) to the root by doing similar kind of operations. Notice that we now have $p_0q_0$ and $(p_0 \oplus p_1)(q_0 \oplus q_1)$ and we need $(p_0q_0 \oplus (p_0 \oplus p_1)(q_0 \oplus q_1))x_n$, which is equivalent to XOR-ing $p_0q_0$ with $(p_0 \oplus p_1)(q_0 \oplus q_1)$ and then concatenating the result with $p_0q_0$ (see Figure 1).

Extracting a bit from a byte is costly. Hence, we use table - lookups to avoid this. Instead of going all the way down to the $n^{th}$ level, we stop at
level \( n - \beta \) and use table lookups to perform multiplication of two \( \beta \) variable polynomials. The value of \( \beta \) is taken to be 3, because the table corresponding to \( \beta = 4 \) becomes very large. We thus pack the polynomials \( p \) and \( q \) in 8-bit arrays and use 8-bit XOR to multiply \( p \) and \( q \).

### 2.3 Further Improvement

One may use \( w \)-bit XOR instead of 8-bit, assuming the architecture allows \( w \)-bit word arithmetic, where \( w = 2^k, k \geq 3 \). The motivation is to save on the number of 8-bit XOR’s. Thus, using one \( w \)-bit XOR, one can save \( 2^{\log_2 w - 3} \) many XOR’s. However, doing it this way one can only go up to \( n - \log_2 w \) level, since, as mentioned in the previous section, maintaining a table of size greater than 3-variables is not feasible. Hence using \( w \)-bit words, involves, an additional task of UNPACKING and PACKING the \( w \)-bit word into bytes so that one can use the 8-bit table lookup. The naive approach to do this, is to copy each \( w \)-bit word into a byte array and basically use the same method to multiply two \( \log_2 w \)-variate polynomials using an 8-bit table lookup. And after multiplication copy back the result into a \( w \)-bit word.

We however, instead of directly copying the \( w \)-bit words to and back from byte arrays, use a constant amount of extra space to get an algorithm which not only saves us the cost of copying but also saves on the number of XOR’s. The idea is to use \( 2^{\log_2 w - 3} \) many \( w \)-bit word masks, say \( M_1, \ldots, M_{2^{\log_2 w - 3}} \) plus an additional temporary variable “temp”, where \( M_l \) contains 1 in the bit positions \( j \cdot 2^{\log_2 w - 1} + k, j \in \{0, 2, 4, \ldots, 2^l - 2\} \) and \( k \in \{0, 1, 2, \ldots, 2^{\log_2 w - 1} - 1\} \) and 0 elsewhere. The \( M_l \)’s actually simulate each level of the tree, corresponding to each \( w \)-bit word. Thus, for example, during the 1st level of the tree for each \( w \)-bit word, \( M_1 \) consists of 1 in the bit positions 0, 1, \ldots, \( 2^{\log_2 w - 1} - 1 \) positions and 0 elsewhere. \( M_1 \) is then AND-ed with the \( w \)-bit word to pick the corresponding \( p_0 \) (here we assume the left-most bit to be our LSB) and is stored in the temporary word “temp”; “temp” is then right shifted by \( 2^{\log_2 w - 1} \) and XOR-ed with the \( w \)-bit word to get the corresponding \( p_0 \oplus p_1 \). Thus after doing this we have \( p_0 \) in the first half of the \( w \)-bit word and \( p_0 \oplus p_1 \) in the second half, which is what we wanted. Hence, for each level we need 3 (1 AND, 1 SHIFT and 1 XOR) \( w \)-bit operations. The PACKING process is the same as that of UNPACKING, except that the masks are used in a reverse order. For table look-ups we use \( 2^{\log_2 w - 3} \) many additional masks, \( B_{i+1}, i \in \{0, 1, 2, \ldots, \} \).
$$2^{\log_2 w - 3} - 1 \right\}$$, to extract the corresponding \( i\)th byte from a \( w\)-bit word, where \( B_{i+1} \) contains 1 in the bit positions \( 8 \times i, 8 \times i + 1, 8 \times i + 2, \ldots, 8 \times i + 7 \) and 0 elsewhere.

Thus, for each level of PACKING and UNPACKING, we need 1 \( w\)-bit AND, 1 SHIFT on a \( w\)-bit word and 1 \( w\)-bit XOR operations. Therefore, for each PACKING and UNPACKING procedure we require \( 3 \cdot (\log_2 w - 3) \) \( w\)-bit operations. Also, for each table look-up we require 2 extra \( w\)-bit operations.

We need 1 AND for extracting a particular byte and 1 SHIFT to bring the value of the extracted \( w\)-bit word within the range of 0 to 255. Since for each \( w\)-bit word we require \( 2^{\log_2 w - 3} \) many table look-ups therefore, for each \( w\)-bit word we need \( 2 \cdot 2^{\log_2 w - 3} \) many \( w\)-bit operations for table look-ups.

### 3 A \( w\)-bit Non-recursive Algorithm

In this section, we summarize our discussion in Section 2 to give a \( w\)-bit non-recursive algorithm, called MultANF\(_w\) (see Algorithm 6). The routine MultANF\(_w\) takes as input \( T, A, B, n, w \), where \( A \) and \( B \) are the corresponding \( w\)-bit word representation of two \( n\)-variate polynomials \((n > \log_2 w \geq 3)\) and \( T_{256 \times 256} \) is a 8-bit table look-up. MultANF\(_w\) multiplies the polynomials \( A \) and \( B \) with the help of table \( T \) and stores the result in \( C \).

To do this, the MultANF\(_w\) routine calls the subroutines “PRE_PROCESS” (Algorithm 1), “UNPACK” (Algorithm 2), “EXTRACT_AND_LOOKUPS” (Algorithm 3), “PACK” (Algorithm 4) and “POST_PROCESS” (Algorithm 5). The subroutine PRE_PROCESS corresponds to the operations while descending down the recursion tree, whereas the subroutine “POST_PROCESS” corresponds to the operations while ascending up the recursion tree. Notice that the subroutine ‘UNPACK” is called twice once each for the \( w\)-bit words \( A[i] \) and \( C[i] \).

The subroutines PACK and UNPACK are the same as PACKING and UNPACKING, as described in the previous section (Section 2.3). The subroutine EXTRACT_AND_LOOKUPS extracts each byte from the \( w\)-bit words \( A \) and \( B \); does the corresponding table lookup and then stores the value returned by the table in the exact byte position of \( C \).

**Algorithm 1: PRE_PROCESS \((A, B, n, i)\)**

Input: \( A, B, n, i \)

for \( j = 0, 1, 2, \ldots, 2^n - 1 \) do

for \( k = 0, 1, \ldots, 2^n - i - 1 \) do

\( A[2^n - i - 1 + j \cdot 2^n - i + k] = A[2^n - i - 1 + j \cdot 2^n - i + k] \oplus A[j \cdot 2^n - i + k] \)

\( B[2^n - i - 1 + j \cdot 2^n - i + k] = B[2^n - i - 1 + j \cdot 2^n - i + k] \oplus B[j \cdot 2^n - i + k] \)

end

end
Algorithm 2: UNPACK $(X, n)$ : Unpacks a $w$-bit word to a byte array.

**Input:** a $w$-bit word $X$; $n := \log_2 w - 3$.

for $i = 0, 1, 2, \ldots, n - 1$ do
  temp = Bit-wise AND of $X$ and $M_{i+1}$
  temp = SHIFT right temp by $2^{n+3-i-1}$ (according to our assumption, the left-most bit is the LSB)
  $X = \text{temp XOR } X$
end

Algorithm 3: EXTRACT_AND_LOOK UP $(X, Y, Z, n)$ : Extracts bytes from $w$-bit words $X$ and $Y$, does a table look-up and stores the result in the corresponding byte of $Z$.

**Input:** $w$-bit words $X, Y, Z$; table $T$; $n := \log_2 w - 3$

for $i = 0, 1, \ldots, 2^n - 1$ do
  if $i = 0$ then $Z := T[X \text{ AND } B_1][Y \text{ AND } B_1]$ end
  else temp := $T[(X \text{ AND } B_{i+1}) \text{ SHIFT left by } i \cdot 2^3 \text{ bits.}][Y \text{ AND } B_{i+1}] \text{ SHIFT left by } i \cdot 2^3 \text{ bits.}$ (According to our assumption the left-most bit is the LSB).
  $Z := \text{temp XOR } Z$
end
end

Algorithm 4: PACK $(Z, n)$ : Packs a $w$-bit word into a byte array.

**Input:** a $w$-bit word $Z$; $n := \log_2 w - 3$.

for $i = n - 1, n - 2, n - 3, \ldots, 0$ do
  temp = Bit-wise AND of $Z$ and $M_{i+1}$
  temp = SHIFT right temp by $2^{n+3-i-1}$ (according to our assumption, the left-most bit is the LSB)
  $Z = \text{temp XOR } Z$
end

**Cost Analysis For MultANF** $w$ : In both Algorithms 1 and 5, the loops run for $2^i \cdot 2^{(n-\log_2 w)-i-1} = 2^{n-\log_2 w-1}$. For each such iteration, we do two $w$-bit XOR’s for PRE_PROCESS and one $w$-bit XOR for POST_PROCESS. Hence, the total number of $w$-bit XOR operations for each PRE_PROCESS and POST_PROCESS call are $2 \cdot 2^{n-\log_2 w-1} = 2^{n-\log_2 w}$ and $2^{n-\log_2 w-1}$, respectively. Also, notice that in MultANF$_w$, PRE_PROCESS and POST_PROCESS are each called $n - \log_2 w$ many times. Therefore, the total number of $w$-bit XOR operations required in the PRE_PROCESS and POST_PROCESS part of MultANF$_w$ is $(n - \log_2 w) \cdot 2^{n-\log_2 w} + (n - \log_2 w) \cdot 2^{n-\log_2 w-1} = 3 \cdot (n - \log_2 w) \cdot 2^{n-\log_2 w-1}$.

As discussed in Section 2.3, for each $w$-bit word we require $3 \cdot (\log_2 w - 3)$ many $w$-bit operations for each call to PACK and UNPACK algorithm and
Algorithm 5: POST_PROCESS \((C, n, i)\)

Input: \(C, n, i\)

for \(j = 0, 1, 2, \ldots, 2^n - 1\) do
  for \(k = 0, 1, \ldots, 2^{n-i-1} - 1\) do
    \(C[2^{n-i-1} + j \cdot 2^{n-i} + k] = C[2^{n-i-1} + j \cdot 2^{n-i} + k] \oplus C[j \cdot 2^{n-i} + k]\)
  end
end

Algorithm 6: MultANF\(_w\) \((T, A, B, C, n, w)\) : A non recursive algorithm to multiply two boolean functions in their ANF’s.

Input: 8-bit Look-up Table \(T\); Two polynomials \(A\) and \(B\); \(C\) for Result; number of variables \(n\); word size \(w\)

Output: \(C := \text{Product of } A \text{ and } B\)

for \(i = 0, 1, 2, \ldots, n - \log_2 w - 1\) do
  PRE_PROCESS\((A, B, n - \log_2 w, i)\)
end

for \(i = 0, 1, 2, \ldots, 2^n - \log_2 w - 1\) do
  UNPACK \((A[i], \log_2 w - 3)\)
  UNPACK \((B[i], \log_2 w - 3)\)
  EXTRACT\_AND\_LOOKUP \((A[i], B[i], C[i], \log_2 w - 3)\)
  PACKING \((C[i], \log_2 w - 3)\)
end

for \(i = n - \log_2 w - 1, n - \log_2 w - 2, n - \log_2 w - 3, \ldots, 0\) do
  POST_PROCESS\((C, n - \log_2 w, i)\)
end

\(2^{\log_2 w - 3}\) many \(w\)-bit operations for table look-ups, plus \(2^{\log_2 w - 3}\) many 8-bit table look-ups. The total number of such \(w\)-bit words is \(2^{n - \log_2 w}\). Also notice that UNPACK is called twice whereas PACK is called once. Therefore, the total cost to multiply two \(n\)-variante polynomial using our \(w\)-bit non-recursive algorithm is:

1. \(2^{\log_2 w - 3} \cdot 2^{n - \log_2 w} = 2^{n - 3}\) many 8-bit table look-up.
2. \(2 \cdot 2^{n - \log_2 w} \cdot 2^{\log_2 w - 3} = 2^n - 2\) many \(w\)-bit operations for table look-ups.
3. \(2^{n - \log_2 w} \cdot (3 \cdot (3 \cdot (\log_2 w - 3))) = 9 \cdot (\log_2 w - 3) \cdot 2^{n - \log_2 w}\) many \(w\)-bit operations for PACKING and UNPACKING.
4. \(3 \cdot (n - \log_2 w) \cdot 2^{n - \log_2 w - 1}\) many \(w\)-bit XOR’s for the PRE_PROCESS and POST_PROCESS.

4 Experimental Results

We present experimental results based on three separate implementations of MultANF\(_w\), with \(w = 8\), \(w = 32\) and \(w = 64\). We have used “C” language for our implementation. To further gain in speed we did some further modifications to our algorithm like using macro calls instead of function calls. The table \(T\) is implemented as one-dimensional array instead of a two-dimensional one. Thus, the entry corresponding to \(T[A[i]][B[i]]\) is now \(T[(A[i] << 8) + B[i]]\). For code
Fast Multiplication of the Algebraic Normal Forms of Two Boolean Functions.

<table>
<thead>
<tr>
<th>n</th>
<th>Average Cycles for 8 bit</th>
<th>Average Cycles for 32 bit</th>
<th>Speedup of 32-bit w.r.t. 8 bit</th>
<th>Average Cycles for 64 bit</th>
<th>Speedup of 64-bit w.r.t. 8-bit</th>
<th>Speedup of 64-bit w.r.t. 32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>408.55</td>
<td>121.01</td>
<td>121.01</td>
<td>92.73</td>
<td>5.98</td>
<td>1.45</td>
</tr>
<tr>
<td>7</td>
<td>1138.23</td>
<td>428.95</td>
<td>199.38</td>
<td>5.71</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2273.35</td>
<td>1032.89</td>
<td>2.02</td>
<td>1022.83</td>
<td>2.22</td>
<td>1.01</td>
</tr>
<tr>
<td>9</td>
<td>5013.86</td>
<td>1853.20</td>
<td>2.02</td>
<td>1022.83</td>
<td>2.22</td>
<td>1.01</td>
</tr>
<tr>
<td>10</td>
<td>11055.29</td>
<td>3871.94</td>
<td>2.65</td>
<td>1022.83</td>
<td>2.22</td>
<td>1.01</td>
</tr>
<tr>
<td>11</td>
<td>23608.47</td>
<td>7711.84</td>
<td>2.65</td>
<td>1022.83</td>
<td>2.22</td>
<td>1.01</td>
</tr>
<tr>
<td>12</td>
<td>53976.73</td>
<td>16093.17</td>
<td>2.65</td>
<td>1022.83</td>
<td>2.22</td>
<td>1.01</td>
</tr>
<tr>
<td>13</td>
<td>109892.07</td>
<td>34223.26</td>
<td>2.65</td>
<td>23296.39</td>
<td>4.46</td>
<td>1.47</td>
</tr>
<tr>
<td>14</td>
<td>221928.42</td>
<td>73352.13</td>
<td>2.65</td>
<td>49992.79</td>
<td>4.44</td>
<td>1.47</td>
</tr>
<tr>
<td>15</td>
<td>466755.57</td>
<td>153265.65</td>
<td>2.65</td>
<td>101450.16</td>
<td>4.60</td>
<td>1.51</td>
</tr>
<tr>
<td>16</td>
<td>1014411.71</td>
<td>321682.42</td>
<td>2.65</td>
<td>212650.40</td>
<td>4.77</td>
<td>1.51</td>
</tr>
<tr>
<td>17</td>
<td>2075710.70</td>
<td>681210.39</td>
<td>2.65</td>
<td>441665.78</td>
<td>4.77</td>
<td>1.51</td>
</tr>
<tr>
<td>18</td>
<td>4041203.98</td>
<td>1430464.38</td>
<td>2.65</td>
<td>915821.38</td>
<td>4.81</td>
<td>1.57</td>
</tr>
<tr>
<td>19</td>
<td>8084408.84</td>
<td>3132142.40</td>
<td>2.65</td>
<td>2509439.46</td>
<td>4.81</td>
<td>1.57</td>
</tr>
<tr>
<td>20</td>
<td>20418478.40</td>
<td>6441914.73</td>
<td>2.65</td>
<td>5114394.99</td>
<td>4.99</td>
<td>1.66</td>
</tr>
<tr>
<td>21</td>
<td>43212647.62</td>
<td>13552825.50</td>
<td>2.65</td>
<td>10069153.25</td>
<td>4.07</td>
<td>1.28</td>
</tr>
<tr>
<td>22</td>
<td>89715956.45</td>
<td>28185683.11</td>
<td>2.65</td>
<td>21062665.54</td>
<td>4.11</td>
<td>1.29</td>
</tr>
<tr>
<td>23</td>
<td>190141764.33</td>
<td>59136265.78</td>
<td>2.65</td>
<td>45559914.11</td>
<td>4.17</td>
<td>1.30</td>
</tr>
<tr>
<td>24</td>
<td>401052397.73</td>
<td>130650693.03</td>
<td>2.65</td>
<td>106224818.55</td>
<td>3.78</td>
<td>1.23</td>
</tr>
<tr>
<td>25</td>
<td>838518957.22</td>
<td>299998311.34</td>
<td>2.65</td>
<td>272976258.05</td>
<td>3.07</td>
<td>1.10</td>
</tr>
<tr>
<td>26</td>
<td>1709215937.18</td>
<td>584223651.94</td>
<td>2.65</td>
<td>609701684.94</td>
<td>2.93</td>
<td>1.08</td>
</tr>
<tr>
<td>27</td>
<td>3635571721.89</td>
<td>1122794840.80</td>
<td>2.65</td>
<td>1239783643.15</td>
<td>2.93</td>
<td>1.07</td>
</tr>
<tr>
<td>28</td>
<td>7543738144.69</td>
<td>1735720452.18</td>
<td>2.65</td>
<td>2541683699.56</td>
<td>2.97</td>
<td>1.04</td>
</tr>
<tr>
<td>29</td>
<td>15906540921.85</td>
<td>5572650229.49</td>
<td>2.65</td>
<td>5109922401.84</td>
<td>3.06</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 1: Table showing the speed (in cycles) comparisons between 8-bit, 32-bit and 64-bit implementations.

All our implementations were run on a HP Z800 Workstation. The machine has 96 GB RAM, 12 Intel(R) Xeon(R) CPU X5675 3.07GHz processor, 384 kB L1 cache, 1536 kB L2 cache and 12288 kB L3 cache. As for OS, we have used "Ubuntu 12.04 LTS" with Linux 3.2.0-24-generic x86_64 kernel version. To get the running time in terms of number of cycles, we have used the “RDTSC” register, available in Intel processors. To train the “cache” and “branch predictors”, we have used one-fourth of the total number of iterations (For further details see Shay Gueron [Gue11]).

Table 1, compares the speed of our three implementations for number of variables “n” ranging from 6 to 30. As expected, our 64-bit implementation works faster than the other two implementations. A single multiplication of 30-variate polynomial using MultANF_{w} can be done in 1.66 secs on an average.

We next compare our 8-bit implementation MultANF_{8} in “C” with that of SAGE. Table 2 gives the comparison of the performance of MultANF_{8} with SAGE. The entries in the tables, denote the running time in seconds (s) and nanoseconds (ns). To get the timings in seconds and nanoseconds we have used the functions “timeit” for SAGE. Same inputs were used for the two

optimization we have used “O1” and the “funroll-all-loops” directives of the “gcc” compiler.
Table 2 Comparison with SAGE. In each case, the timings are averaged over 1000 runs.

<table>
<thead>
<tr>
<th>n</th>
<th>MultANF</th>
<th>sage</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.80 ns</td>
<td>94773.05 ns</td>
</tr>
<tr>
<td>4</td>
<td>1.84 ns</td>
<td>127928.97 ns</td>
</tr>
<tr>
<td>5</td>
<td>55.55 ns</td>
<td>197319.98 ns</td>
</tr>
<tr>
<td>6</td>
<td>70.78 ns</td>
<td>354038.95 ns</td>
</tr>
<tr>
<td>7</td>
<td>161.31 ns</td>
<td>762128.12 ns</td>
</tr>
<tr>
<td>8</td>
<td>718.90 ns</td>
<td>1700400.83 ns</td>
</tr>
<tr>
<td>9</td>
<td>799.88 ns</td>
<td>3205485.06 ns</td>
</tr>
<tr>
<td>10</td>
<td>1444.70 ns</td>
<td>7070338.01 ns</td>
</tr>
<tr>
<td>11</td>
<td>1511.90 ns</td>
<td>14413833.62 ns</td>
</tr>
<tr>
<td>12</td>
<td>1572.56 ns</td>
<td>32455171.93 ns</td>
</tr>
<tr>
<td>13</td>
<td>1514.18 ns</td>
<td>69974661.11 ns</td>
</tr>
<tr>
<td>14</td>
<td>1528.48 ns</td>
<td>162449117.1 ns</td>
</tr>
<tr>
<td>15</td>
<td>17486.74 ns</td>
<td>336447669.9 ns</td>
</tr>
</tbody>
</table>

4.1 Multiplying Sparse Polynomials

For sparse implementation, a monomial is represented by a $\delta$-bit word and two polynomials are given as two arrays $A$ and $B$ of monomials. Multiplication of two polynomials corresponds to the bit-wise OR of the corresponding $\delta$-bit words. Suppose we want to multiply two sparse polynomials $p$ with $l_p$ monomials and $q$ with $l_q$ monomials. For our sparse implementation (let us call it the quadratic implementation), we take the input arrays $A$ and $B$ and OR every element of array $A$ with that of array $B$, and store them in another array $C$. The array $C$ is then sorted using a non recursive (the process stack is simulated internally) implementation of randomized quick sort. Repetitions are removed by either deleting the monomial (if its number of repetitions is even) or replacing all the entries by just one entry (if the number of repetitions is odd).

Experiments were done to compare the speeds of SAGE for sparse polynomials with that of quadratic implementation. The experimental results not only show that the algorithm used by SAGE is slower than the quadratic implementation but also suggests that the SAGE algorithm depends both on the sizes of $A$ and $B$ (i.e., $l_p$ and $l_q$) and the number of variables involved. But
the quadratic implementation only depends on \( l_p \) and \( l_q \). For example to multiply two polynomials each with 1000 monomials Sage took 7.43 seconds for \( n = 30 \) and 34 seconds for \( n = 63 \), whereas for the quadratic implementation it took 0.17 seconds for both \( n = 30 \) and \( n = 63 \).

Based on experimental results, we also found that if \( l_p l_q < 2^n - \alpha \), then the quadratic algorithm performs better than MultANF\(_{2^\alpha}\), where \( \alpha = 3, 5, 6 \).

5 Conclusion

In this paper we have proposed a new non-recursive algorithm MultANF\(_w\), which multiplies two Boolean functions in their ANF’s. MultANF\(_w\) tries to use the \( w \)-bit word arithmetic, if the architecture supports it. With this in mind, three variants of MultANF\(_w\) are proposed for \( w = 8, 32 \) and 64. We show that the 64-bit implementation is better than the other two. A detailed comparison of MultANF\(_w\) with a sparse implementation tells us, when one should switch from the sparse implementation to the dense implementation, i.e., MultANF\(_w\). Lastly, a comparison between our implementations (sparse and dense implementations) with that of the software package Sage shows that, our implementations are faster than Sage.

The MultANF\(_{64}\) algorithm is used to symbolically compute TRIVIUM. This is still a work in progress. We wish to do a thorough structural analysis of the output polynomials of TRIVIUM and conduct different randomness tests on it.

References


[BGTZ08] Richard P. Brent, Pierrick Gaudry, Emmanuel Thomé, and Paul Zimmermann. Faster multiplication in \( \text{GF}(2)\)[x]. In van der Poorten and Stein [vdPS08], pages 153–166.


On canonical forms of ring-linear codes

Thomas Feulner

Abstract This work presents an algorithm for the computation of canonical forms of linear codes over arbitrary finite chain rings under the action of the linear isometry group. It is a generalization of a previous work by the author on the canonization of linear codes over finite fields. The algorithm is based on the ideas of partition refinements and it will compute the automorphism group of the code as a byproduct.

Keywords automorphism group · canonical form · finite chain ring · linear code · isometry · partition refinement

Mathematics Subject Classification (2000) 05E20 · 20B25 · 94B05

1 Introduction

A ring $R$ is said to be a left chain ring if the set of left ideals of $R$ is linearly ordered with respect to inclusion. Right chain rings are defined analogously. A finite ring $R$ is a left chain ring if and only if it is a right chain ring. Since we will only consider the finite case, we just use the term chain ring in the following. A linear code $C$ of length $n$ over the chain ring $R$ is a submodule of the left $R$-module $R^n$.

For coding theoretic purposes different distances $d$ can be defined on $R^n$. We state the most prominent examples without giving their exact definition since the distances themselves become irrelevant in the following:

- the Hamming distance $d_{Ham}$ and the homogeneous distance $d_{hom}$ is defined for arbitrary chain rings,
- the Lee distance $d_{Lee}$ and Euclidean distance $d_{Euc}$ is defined over $\mathbb{Z}_p^n$.

This work is supported by DFG priority program SPP 1489 under grant WA-1666/7-1

T. Feulner
University of Bayreuth
E-mail: thomas.feulner@uni-bayreuth.de
A (left-)linear $d$-isometry $t : C \rightarrow R^n$ is a (left-)linear mapping preserving the distance $d$. The extension theorem of MacWilliams was generalized by Wood [10], [11] and many others to several classes of chain rings and distances defined on them. It states that the equivalence classes of (left) linearly $d$-isometric linear codes coincide with orbits of (left) $U_d$-monomial transformations for suitable subgroups $U_d$ of the group of units $R^\times$ of $R$.

The group of (left) $U_d$-monomial transformations is isomorphic to the wreath product $U_d \wr S_n := \{ (\varphi; \pi) \mid \varphi \in (U_d)^n, \pi \in S_n \}$ where the multiplication of $(\varphi; \pi), (\psi; \sigma) \in U_d \wr S_n$ is defined by

$$(\varphi; \pi)(\psi; \sigma) := \left( (\varphi_0\psi_{-1}(0), \ldots, \varphi_{n-1}\psi_{-1}(n-1)); \pi\sigma \right)$$

and the action on $c \in R^n$ is given by

$$(\varphi; \pi)(c_0, \ldots, c_{n-1}) := \left( \epsilon_{-1}(0)\varphi_0^{-1}, \ldots, \epsilon_{-1}(n-1)\varphi_{n-1}^{-1} \right).$$

In this paper, we will restrict ourselves to those distances $d$ where the extension theorem of MacWilliams holds and $U_d = R^\times$. Theorem 9.4 in [11] gives an easy characterization of those distances and shows that the Hamming distance as well as the homogeneous distance have this property. This fact allows us to identify the equivalence classes of left linearly $d$-isometric codes by orbits under the group action of $R^\times \wr S_n$.

Our goal is to give a practical algorithm which solves the canonization problem for this group action in the following sense: Let $G$ be a group acting on a set $X$ from the left and let $\mathcal{L}(G)$ denote the set of subgroups of $G$.

**Problem 1 (Canonization)** Compute a function

$$\text{Can}_G : X \rightarrow X \times G \times \mathcal{L}(G)$$

$$x \mapsto (\text{CF}_G(x), \text{TR}_G(x), \text{Stab}_G(x))$$

with

$$\forall x \in X, \forall g \in G : \text{CF}_G(x) = \text{CF}_G(gx) \quad \text{(canonical form)}$$

$$\forall x \in X : \text{TR}_G(x)x = \text{CF}_G(x) \quad \text{(transporter element)}$$

$$\forall x \in X : \text{Stab}_G(x) = \{ g \in G \mid gx = x \} \quad \text{(stabilizer)}$$

The element $\text{CF}_G(x)$ is called the canonical form of $x$ and the element $\text{TR}_G(x) \in G$ a transporter element. The group element $\text{TR}_G(x)$ is well-defined up to the multiplication with the stabilizer $\text{Stab}_G(x)$ of $x$ from the right.

In our case the group and the set is finite, hence a standard orbit-stabilizer algorithm would already solve Problem 1. On the other hand we have to expect huge orbit sizes already for small parameters, which implies that this approach will certainly fail.

The structure of this extended abstract is as follows. First we will give a short summary of the main ideas on the canonization of linear codes over finite fields given in [1]. Section 3 will give some basic facts on chain rings and linear codes defined on them. Furthermore in Subsection 3.2, we will rephrase the group action of $R^\times \wr S_n$ such that we are able to similarly work with generator matrices instead of submodules.
Finally in Section 4, we give our main ideas for an efficient canonization algorithm for linear codes over finite chain rings, i.e. an algorithm which solves Problem 1 in this special case. Compared to a pure test on equivalence of two linear codes as described in [5] this algorithm is also practical when one has to identify the equivalence class of a given linear code in a huge collection of representatives, for example in some database or when one has to reject $d$-isometric copies in some classification algorithm.

2 The canonical form algorithm for linear codes over finite fields revisited

This section should give a quick overview on the ideas applied in [1] for the computation of a canonical form of a linear code under the action of the $F^*$-monomial group.

First of all we notice that we are allowed to restrict the action to subspaces of some fixed dimension $k$ since the dimension is preserved by linearity. Furthermore, we can replace the linear code $C$ by the orbit of $GL_k(F_q)$ on a generator matrix $\Gamma$ of $C$. This observation allows us to consider the group action of $GL_k(F_q) \times (F^n_q \rtimes S_n) = (GL_k(F_q) \times F^n_q) \rtimes S_n$
onumber

on the set $F^{k \times n \times k}_q$ of $k \times n$ matrices of full row rank $k \leq n$ instead.

Now let $G = GL_k(F_q) \times F^n_q$ denote the normal subgroup of this semidirect product and let $F^{k \times n \times k}_q / G := \{ G \Gamma | \Gamma \in F^{k \times n \times k}_q \}$

be its set of orbits. The symmetric group naturally acts on $F^{k \times n \times k}_q / G$ via $\pi \cdot G \Gamma := G(\pi \Gamma)$. Since the sets $(G \rtimes S_n) \Gamma$ and $\bigcup_{\pi \in S_n} \pi \cdot G \Gamma$ are equal this observation leads to the following constructive definition of a canonization algorithm

\[
\text{Can}_{G \rtimes S_n} : F^{k \times n \times k}_q \rightarrow F^{k \times n \times k}_q \times (G \rtimes S_n) \times (G \rtimes S_n)
\]

\[
\Gamma \mapsto (\text{CF}_{G \rtimes S_n} (\Gamma), \text{TR}_{G \rtimes S_n} (\Gamma), \text{Stab}_{G \rtimes S_n} (\Gamma))
\]

which we will discuss in the following:

- Compute some permutation $\pi_0 := \text{TR}_{S_n} (G \Gamma) \in S_n$ such that $\pi_0 G \Gamma = \text{CF}_{S_n} (G \Gamma)$ is a canonical form for the group action of $S_n$ on $F^{k \times n \times k}_q / G$, i.e. solve the canonization problem for $S_n$ acting on $F^{k \times n \times k}_q / G$.

- Define a canonization algorithm $\text{Can}_G = (\text{CF}_G, \text{TR}_G, \text{Stab}_G)$ for the group action of $G$ on $F^{k \times n \times k}_q / G$.

- Define $\text{Can}_{G \rtimes S_n}$ via:

  1. $\text{CF}_{G \rtimes S_n} (\Gamma) := \text{CF}_G (\pi_0 \Gamma)$
  2. $\text{TR}_{G \rtimes S_n} (\Gamma) := (\text{TR}_G (\pi_0 \Gamma), \pi_0)$
  3. $\text{Stab}_{G \rtimes S_n} (\Gamma)$ is generated by $\text{Stab}_G (\Gamma)$ and the group elements $(\text{TR}_G (\pi_0 \Gamma), \pi_0)^{-1} (\text{TR}_G (\pi_0 \pi_1 \Gamma), \pi_0 \pi_1), \forall \pi_1 \in \text{Stab}_S (G \Gamma)$

\footnote{For simplicity, we restrict our formulation to linear isometry classes.}
For the action of the group \( G = \text{GL}_k(\mathbb{F}_q) \times \mathbb{F}_q^n \) on the set \( \mathbb{F}_q^{k \times n,k} \) we are able to give an efficient canonization algorithm which is hardly more difficult than the computation of the row reduced echelon form of the matrix \( \Gamma \). Indeed, in the binary case both algorithms coincide. In fact, this algorithm returns the minimal element in the orbit if one considers matrices as lexicographically ordered sequences of co-lexicographically ordered vectors. This will become important later.

The function \( \text{Can}_n \) is computed via some refinement-based backtrack search algorithm which is well-known from the computation of a canonical labeling of a graph, cf. [7] and other combinatorial objects, see for example [6]. We use a more group theoretic description given in [3, Section 5.6] to define the search tree. Nevertheless we start with some basic definitions on partitions of the set \( [n] := \{0, \ldots, n - 1\} \):

A partition of \( [n] \) is an ordered sequence \( p = (P_0, \ldots, P_{l-1}) \) of disjoint nonempty subsets of \([n]\) whose union is equal to \([n]\). We call the subsets \( P \in p \) cells of the partition. Cells of cardinality 1 are singletons and the partition \( p \) is discrete if all its cells are singletons. If all cells of a partition \( p \) are intervals we call \( p \) a standard partition.

In the following we will always use upper-case letters for standard partitions. The stabilizer

\[
S_p := \text{Stab}_n(p) := \bigcap_{P \in p} \text{Stab}_n(P)
\]

of the (standard) partition \( p \) is a (standard) Young subgroup of \( S_n \). With \( \text{Fixed}(p) := \{ i \in [n] \mid \{ i \} \in p \} \) we refer to those indices which define singletons of \( p \), i.e. fixed points under the group action of \( S_p \) on \([n]\).

In [7] the ordered partition \( p = (P_0, \ldots, P_{l-1}) \) of \([n]\) is used as a data type to store the set of permutations

\[
S_p^\pi := \left\{ \pi \in S_n \mid \forall j \in [l], \forall i \in \left[ \sum_{k=0}^{j-1} |P_k| \right] \cup \left[ \sum_{k=0}^{j-1} |P_k| + 1, \ldots, \sum_{k=0}^{j} |P_k| \right] : \pi^{-1}(i) \in P_j \right\}.
\]

This set \( S_p^\pi \) is equal to the coset \( S_p^\pi \) for \( \pi \in S_p^\pi \) and the standard Young subgroup \( S_\pi \) which is defined by the standard partition \( \pi \) whose number of cells and sequence of cell sizes is equal to \( p \). Vice versa, there is always some ordered partition \( p \) which represents the coset \( S_\pi P \) of some standard Young subgroup \( S_\pi \).

**Definition 1** We call a partition \( p \) finer than the partition \( p' \) if \( \forall \pi \in S_p \) is a subset of \( S_{p'} \). We also call \( p \) a refinement of \( p' \) and say that \( p' \) is coarser than \( p \).

Analogously to [3], we compute \( \text{Can}_{n_0}(G\Gamma) \) for some given element \( G\Gamma \in \mathbb{F}_q^{k \times n,k} / G \) by carrying out a backtrack search on the group \( S_n \). We describe this by defining a search tree \( T(G\Gamma, S_n) \) using the following two basic operations

**Partitioning Rule:** Take a coset \( S_\pi \pi' \) of a standard Young subgroup and refine it to a subcoset \( S_\pi' \sigma' \pi \) with \( \pi' \preceq \pi \) and \( \sigma \in S_\pi \). In practice, the refinement rule depends on the input \( G\Gamma \), but we require that for some other search tree \( T(\pi_0 G\Gamma, S_n) \) with \( \pi_0 \in S_n \) the result for the coset \( S_\pi \pi \pi_0^{-1} \) must be \( S_\pi' \sigma \pi \pi_0^{-1} \).
In practice, we will derive the refinement transformation by the application of the Homomorphism Principle [4]: Suppose that there is a fixed function \( f_{\mathcal{D}} : \mathbb{P}_q^{k \times n,k} \to X^n \) from the set of orbits of \( G \) on \( \mathbb{P}_q^{k \times n,k} \) onto some set \( X^n \) (with the natural action of \( S_{2^k} \) on \( n \)-tuples) which is compatible\(^2\) with the action of \( S_{2^k} \). The function \( f_{\mathcal{D}} \) is called an \( S_{2^k} \)-homomorphism. Then we define

- the group element \( \sigma \in S_{2^k} \) by cell-wise sorting the entries of the vector \( f_{\mathcal{D}}(\pi G^\Gamma) \) lexicographically and
- the subgroup \( S_{\mathcal{D}} := \text{Stab}_{S_{\mathcal{D}}} \left( \sigma f_{\mathcal{D}}(\pi G^\Gamma) \right) \) to be the stabilizer of the permuted image.

**Individualization:** Take a coset \( S_{\mathcal{D}} \pi \neq \{\pi\} \) of a non-trivial standard Young subgroup and partition it into \( \{S_{\mathcal{D}} \rho_0 \pi, \ldots, S_{\mathcal{D}} \rho_{n-1} \pi\} \) where \( S_{\mathcal{D}} := \text{Stab}_{S_{\mathcal{D}}} (i) \) is the stabilizer of the point \( i = \min(P_j) \) for some \( P_j \in \mathcal{D} : |P_j| > 1 \) and \( \{\rho_0, \ldots, \rho_{n-1}\} \) is a right transversal for \( S_{\mathcal{D}} \) in \( S_{2^k} \). Again the choice of the cell \( P_j \) may depend on \( X \) but has to be equal for isomorphic situations, i.e. we will pick the same cell \( P_j \) for the coset \( S_{\mathcal{D}} \pi \) occurring in the search tree \( T(\pi_0^{-1} G^\Gamma, S_n) \).

Altogether, these operations define the search tree \( T(G^\Gamma, S_n) \) in the following way:

1. The root node of \( T(G^\Gamma, S_n) \) is defined by a refinement of \( S_n \) and we associate the empty sequence \( \Phi(G^\Gamma, S_{\mathcal{D}} \pi) := (\) as an evaluation to it,
2. if \( S_{\mathcal{D}} \pi \neq \{\pi\} \) is a node of \( T(G^\Gamma, S_n) \) with evaluation \( \Phi(G^\Gamma, S_{\mathcal{D}} \pi) = (v_0, \ldots, v_{l-1}) \), then the child nodes of \( S_{\mathcal{D}} \pi \) are the refinements \( \{S_{\mathcal{D}} \rho_0 \pi, \ldots, S_{\mathcal{D}} \rho_{n-1} \pi\} \) of the cosets \( \{S_{\mathcal{D}} \rho_0 \pi, \ldots, S_{\mathcal{D}} \rho_{n-1} \pi\} \) into which \( S_{\mathcal{D}} \pi \) is partitioned by the partitioning rule. To the child node \( S_{\mathcal{D}} \rho_i \pi \) we associate the evaluation \( \Phi(G^\Gamma, S_{\mathcal{D}} \rho_i \pi) := (v_0, \ldots, v_{l-1}, f_{\mathcal{D}}(\rho_i \pi)) \).

This search tree is now used to define the canonical form \( \text{CF}_{S_n}(G^\Gamma) \), the transporter element \( \text{TR}_{S_n}(G^\Gamma) \) and the stabilizer of \( G^\Gamma \) under the action of the symmetric group in the usual way:

- Each leaf node \( \{\pi\} \) defines a single permutation \( \pi \in S_n \).
- We define the canonical form \( \text{CF}_{S_n}(G^\Gamma) \) of \( G^\Gamma \) to be the minimum among all elements \( \pi G^\Gamma \) where \( \{\pi\} \) is a leaf node of \( T(G^\Gamma, S_n) \) possessing a minimal evaluation. We store the corresponding permutation to be the transporter element \( \text{TR}_{S_n}(G^\Gamma) \).
- The stabilizer \( \text{Stab}_{S_n}(G^\Gamma) \) is a subset of \( \{\pi \in S_{2^k} \mid (\{\pi\} \text{ is a leaf in } T(G^\Gamma, S_n) \text{ with minimal evaluation}\} \}

Obviously, the evaluation of the nodes of \( T(G^\Gamma, S_n) \) could be used to define a pruning operation on \( T(G^\Gamma, S_n) \): If \( \Phi(G^\Gamma, S_{\mathcal{D}} \pi) < \Phi(G^\Gamma, S_{\mathcal{D}} \sigma) \) for two distinct nodes \( S_{\mathcal{D}} \pi \) and \( S_{\mathcal{D}} \sigma \) of \( T(G^\Gamma, S_n) \) then we can discard the subtree in \( S_{\mathcal{D}} \pi \sigma \) since it does not contain a leaf node with minimal evaluation.

Furthermore, if we define \( f_{\mathcal{D}}(G^\Gamma) := \text{Can}_G(\Gamma) \) for the discrete standard partition \( \mathcal{D} \) of \( [n] \) then the leaf nodes of \( T(G^\Gamma, S_n) \) will be in bijection with the elements needed for the definition of \( \text{Can}_{G \times S_n} \) as defined in Equation (2).

\(^2\) \( f_{\mathcal{D}}(\pi G^\Gamma) = \pi f_{\mathcal{D}}(G^\Gamma) \) for all \( \pi \in S_{2^k} \) and all \( \Gamma \in \mathbb{P}_q^{k \times n,k} \)
Finally, we would like to emphasize that we can also choose other codomains for the $S_P$-homomorphisms $f_P$ applied in the Homomorphism Principle [4] as long as we are able to guarantee that the preimages of the stabilizers will be standard Young subgroups. In particular, we can use a sequence of such homomorphisms for iterative refinements, cf. the canonical form algorithm for graphs [7]. Furthermore, if the action on the codomain is trivial we can still use the resulting value for the pruning mechanism as we will see in the following.

**Definition 2** We will access submatrices of $\Gamma \in \mathbb{F}_q^{k \times n}$ in the following way:

- $\Gamma_{*, j}$ denotes the $i$-th column of $\Gamma$. Similarly, we write $\Gamma_{i, *}$ for the $i$-th row.
- For a sequence $I := (i_0, \ldots, i_{m-1})$ of indices $i_j \in [n]$ we write
  $$\Gamma_{*, I} := (\Gamma_{*, i_0}, \ldots, \Gamma_{*, i_{m-1}})$$
  for the projection of the matrix onto the columns given by $I$.

**Definition 3** Let $F$ be an injective tuple of elements in $[n]$. We call a matrix $\Gamma$ an $F$-semicanonical representative of the orbit $G\Gamma$ if $\Gamma_{*, F} = \mathbb{C} F_{\text{GL}_k(F_q) \times F^*}$ where $\Gamma_{*, F}$ is an $F\Gamma$-semicanonical representative of $G\Gamma$ and the action on the codomain is trivial. Additionally, if one arranges $F$ in such a way that it corresponds to the order of the indices getting fixed on the path from the root node to the actual node, one can reuse the results of the predecessor, i.e. one has to minimize all newly fixed column vectors added in the individualization step and the refinement process of this node under the action of the stabilizer $\text{Stab}_{\text{GL}_k(F_q) \times F^*} \left( \Gamma_{*, F} \right)$. The process of computing $F$-semicanonical representatives was also called the inner minimization in [1].

Further $S_P$-homomorphisms can be computed by making use of a set of code-words of some given weights. We will omit the details on this method, the idea can already be found in [5].

A natural choice for the function $f_P$ which does not lead to a refinement but which will give a strong evaluation during the backtracking is now given by a fixed arrangement $F$ of the elements in $\text{Fixed}(P)$:

We define

$$f_P : \mathbb{F}_q^{k \times n, k} / G \rightarrow \mathbb{F}_q^{k \times |F|}$$

$$\Gamma \mapsto \left( \Gamma^{(F)} \right)_{*, F}$$

where $\Gamma^{(F)}$ is an $F$-semicanonical representative of $G\Gamma$ and the action on the codomain is trivial. Additionally, if one arranges $F$ in such a way that it corresponds to the order of the indices getting fixed on the path from the root node to the actual node, one can reuse the results of the predecessor, i.e. one has to minimize all newly fixed column vectors added in the individualization step and the refinement process of this node under the action of the stabilizer $\text{Stab}_{\text{GL}_k(F_q) \times F^*} \left( \Gamma^{(E, \pi)}_{*, F} \right)$. The process of computing $F$-semicanonical representatives was also called the inner minimization in [1].

Further $S_P$-homomorphisms can be computed by making use of a set of code-words of some given weights. We will omit the details on this method, the idea can already be found in [5].

Although we gave a description of the search tree which favors a breadth-first-search traversing order this tree will be built in a depth-first-search manner. This is motivated by the existence of further pruning options which are based on the group of known automorphisms of the code. Details can be found in [1, Subsection 5.2].
Example 1 Let $\Gamma := \begin{pmatrix} 1 & 0 & 1 & \alpha^2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ be the generator matrix of a linear code over $\mathbb{F}_4 := \{0, 1, \alpha, \alpha^2\}$. Without any refinements applied in the generation of $T(\Gamma, S_4)$ this tree would have 24 leafs each corresponding to one permutation $\pi \in S_n$.

The projection of a $(0,1,2)$-semicanonical representative of the orbit $G\pi\Gamma$ for $\pi \in S_n$ to its first three columns is always equal to $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$, since three arbitrarily chosen column vectors are linearly independent. Hence, all non leaf nodes on the same level in $T(\Gamma, S_4)$ will carry an equal evaluation and we are not able to prune any of them by this operation. The $(0,1,2,3)$-semicanonical representative $\text{Can}_G(\pi\Gamma)$ of the orbit $G\pi\Gamma$ is either equal to $\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ or $\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$. The evaluation function partitions the leaf nodes of $T(\Gamma, S_4)$ into two classes each containing 12 elements.

We can choose the permutation $\pi_0 := TR_{S_4}(\Gamma) := (2,3)$ to be the transporter element for the canonization of $\Gamma$. Hence, we set

1. $\text{CF}_{G \rtimes S_n}(\Gamma) := \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

2. $\text{TR}_{G \rtimes S_n}(\Gamma) := (\text{TR}_{G}(\pi_0\Gamma), (2,3)) = \left( \left( \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix}, (\alpha, 1, 1, 1) \right), (2,3) \right)$

3. $\text{Stab}_{G \rtimes S_n}(\Gamma)$ is generated by $\text{Stab}_G(\Gamma) = \left\{ \left( \begin{pmatrix} \phi & 0 \\ 0 & \phi \end{pmatrix}, (\phi, \phi, \phi, \phi) \right) \mid \phi \in \mathbb{F}_4^* \right\}$

and the group elements

$$(\text{TR}_{G}(\pi_0\Gamma), \pi_0)^{-1}(\text{TR}_{G}(\pi_0\pi_1\Gamma), \pi_0\pi_1), \forall \pi_1 \in \text{Stab}_{S_4}(\Gamma)$$

Altogether this short description shows that this partition refinement approach will generalize to arbitrary chain rings as soon as we are able to give an efficient inductive algorithm for the computation of $\text{CF}_{G^{\lambda,\mu}}(\Gamma)$ for the inner group action of a group $G^{\lambda,\mu}$ on generator matrices $\Gamma$, for the definition of both objects see Section 3.2. The definition of other refinements in the backtracking follows standard arguments and will not be part of this work. Similarly, the pruning by automorphism only depends on the permutational part of the automorphism groups and hence is similarly applicable.

3 Codes over Chain rings

3.1 Chain rings

In order to define the notion of a basis and generator matrix of a linear code, we first give some elementary results on chain rings, see for example [2] or [8]. The Jacobson radical of the chain ring $R$ will be denoted by $\text{Rad}(R)$. It is the unique maximal ideal in $R$. Suppose $\theta$ is some arbitrarily chosen element of $\text{Rad}(R) \setminus \text{Rad}(R)^2$. Any proper ideal $I$ of $R$ is a principal ideal and there is some positive integer $\lambda$ but with the help of the group of known automorphisms and [1, Subsection 5.2].
Every chain ring has a Teichmüller coordinate set. The element \( a \) defined coefficient \( \theta \) is uniquely defined up to conjugation with elements in \( R^{*} \). Within this work the ring \( R \) is called a Teichmüller coordinate set of \( R \) if \( 0 \in T \) and \( T^* := T \setminus \{0\} \) is a multiplicatively closed set of representatives of \( (R/Rad(R))^* \cong \mathbb{F}_q^{*} \). Every chain ring has a Teichmüller coordinate set.

If \( R \) is a commutative chain ring, the Teichmüller coordinate set of \( R \) is unique. Otherwise, these sets are uniquely defined up to conjugation with elements in \( R^{*} \). With the help of a fixed Teichmüller coordinate set \( T \), any element \( a \in R \) has a uniquely defined \( \theta \)-adic decomposition:

\[
a = \sum_{i=0}^{m-1} a_i \theta^i, \quad \text{with } a_i \in T.
\]

For \( x \in [m] \) let \( \text{coeff}^{(x)} : R \to T \) denote the projection of a ring element \( a \) to its uniquely defined coefficient \( a_x \) in its \( \theta \)-adic decomposition \( a = \sum_{i=0}^{m-1} a_i \theta^i \).

Within this work the ring \( R \) as well as its integer parameters \( m, q, p, r \) are fixed. The element \( \theta \in R \) should be a fixed generator of \( Rad(R) \) and \( \xi \) a fixed generator of \( T^* \) for a Teichmüller coordinate set \( T \) of \( R \). Finally by [8], there is a unique power \( \tau^r \) of the Frobenius automorphism \( \tau \in \text{Aut}(\mathbb{F}_q) \) such that \( \text{coeff}^{(1)}(\theta \xi) = \tau^r \left( \text{coeff}^{(1)}(\xi \theta) \right) \).

3.2 Linear codes over chain rings

In the following we will summarize some basic facts on linear codes over chain rings needed in this work, see [2] and [12] for more details.

A finite left \( R \)-module \( M \) is decomposable into a direct sum of cyclic modules

\[
M \simeq \bigoplus_{j=0}^{k-1} R/\text{Rad}(R)^{\lambda_j} \simeq \bigoplus_{i=0}^{m-1} (\text{Rad}(R)^i)^{k_i}
\]

with \( 1 \leq \lambda_j \leq m \) and \( k_i := |\{ j \in [k] | m - \lambda_j = i \}| \) and without loss of generality we can assume that the sequence \( \lambda = (\lambda_0, \ldots, \lambda_{k-1}) \) is decreasingly ordered.

Therefore we can define the rank of \( M \) to be \( \text{rk}(M) = k \) and we call \( \lambda \) the shape of the module. The corresponding sequence \( k_\lambda := (k_0, \ldots, k_{m-1}) \) is called the type of the module. A module of shape \( (m, \ldots, m) \) is free.

A sequence \( x_0, \ldots, x_{k-1} \) of elements of the module \( M \) is independent if any \( R \)-linear combination \( \sum_{j=0}^{k-1} a_j x_j = 0 \) of these elements imply \( a_j x_j = 0, \forall j \in \{0, \ldots, k-1 \} \). Any independent set of generators of \( M \) which does not contain \( 0 \in M \) is called a basis of \( M \). The cardinality of any basis of \( M \) is equal to the rank of the module. The preimage of the isomorphism defined by Equation (3) shows the existence of a basis of the module \( M \).
The smallest non-negative integer period(\(x\)) := \(\mu \leq m\) such that \(\theta^\mu x = 0\) is called the period of \(x \in M\). We say that a sequence of basis elements – or for short the basis – is ordered if the corresponding sequence of periods is decreasing. It corresponds to the shape of the module.

**Definition 4** Any \((k \times n)\)-matrix \(\Gamma\) whose rows form an ordered basis of the linear code \(C\) of length \(n\) and rank \(k\) is called generator matrix of \(C\). The set of all generator matrices, which generate a code of shape \(\lambda\) and length \(n\) will be denoted by \(R^{k \times n}\).

We will also say that these generator matrices have shape \(\lambda\).

Furthermore, we call a generator matrix \(\Gamma\) **systematic** if
- the first nonzero entry of a row \(\Gamma_{i,*}\) is equal to \(\theta^{m-p_i}\) where \(p_i := \text{period}(\Gamma_{i,*})\).
  These elements are called pivots.
- The pivot element of a row \(\Gamma_{i,*}\) is further to the right than the pivot element of the row \(\Gamma_{i-1,*}\) for all \(i > 0\).
- All elements above a pivot element \(\Gamma_{i,j}\) are reduced modulo \(\theta^{m-p_i}\) which means that their coefficients in the \(\theta\)-adic decomposition are zero for all indices \(x \geq m-p_i\).

**Example 2** Let \(C \leq \mathbb{Z}_3^2\) with systematic generator matrix \(\Gamma_0\).

\[
\begin{align*}
\Gamma_0 &= \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \\
\Gamma_1 &= \begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix} \\
\Gamma_2 &= \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix} \\
\Gamma_3 &= \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}
\end{align*}
\]

The matrix \(\Gamma_1\) is also a generator matrix of \(C\), but it is not systematic. \(\Gamma_2\) also generates \(C\), but the rows do not define a basis. The rows of \(\Gamma_3\) form a basis, but it is not ordered. The shape of \(C\) is \((2,1)\).

**Fact 2** Not every linear code over a chain ring \(R\) has a systematic generator matrix, but there is at least one permutational equivalent code possessing one.

In this work, we suppose that there is no coordinate \(i \in \{0, \ldots, n-1\}\) with \(c_i = 0, \forall c \in C\). This is a legitimate assumption. However, if we are interested in the canonical form of such a code, we just will remove the zero columns of a generator matrix and append the same number of zero columns to the calculated canonical form. The extension of the automorphism group will also be easy.

In the case of a finite field \(\mathbb{F}_q\), the set of all generator matrices of a code \(C\) of rank \(k\) forms an orbit under the group action of \(GL_k(\mathbb{F}_q)\) on the set of \((k \times n)\)-matrices of full row rank. As we have seen in the example, this is not true for generator matrices of non-free codes.

**Fact 3** The set \( \text{GL}_k(\mathbb{R})\) of block matrices

\[
\begin{pmatrix}
A^{(0,0)} & A^{(0,1)} & \cdots & A^{(0,m-1)} \\
\theta A^{(1,0)} & A^{(1,1)} & \cdots & A^{(1,m-1)} \\
\theta^2 A^{(2,0)} & \theta A^{(2,1)} & \cdots & A^{(2,m-1)} \\
\vdots & \vdots & \ddots & \vdots \\
\theta^{m-1} A^{(m-1,0)} & \theta^{m-2} A^{(m-1,1)} & \cdots & A^{(m-1,m-1)}
\end{pmatrix}
\]

where \(k_\lambda = (k_0, \ldots, k_{m-1})\) and \(A^{(i,j)} \in R^{k_i \times k_j}\), \(A^{(i,j)} \in \text{GL}_{k_i}(\mathbb{R})\) forms a subgroup of \(\text{GL}_k(\mathbb{R})\).
Proposition 1  The group \( GL_{\lambda}(R) \) acts on the set of generator matrices \( R^{k \times n} \) via left multiplication. If \( \Gamma \) is a generator matrix of the linear code \( C \), then the set of all generator matrices of \( C \) is equal to the orbit \( (GL_{\lambda}(R)) \Gamma \).

Proof  [12, Satz 2.22]

Proposition 2  The group action of \( GL_{\lambda}(R) \) on the set of generator matrices \( R^{k \times n} \) is not faithful for non-free codes, i.e.

\[
\{I_k\} \neq N_{\lambda}(R) := \bigcap_{\Gamma \in R^{k \times n}} (GL_{\lambda}(R))_{\Gamma},
\]

\[
= I_k + \left\{ A \in R^{k \times k} \left| A_{i,j} \in \text{Rad}(R)^{\lambda}, \forall i, j \in [k] \right. \right\}
\]

Proof  [12, Lemma 2.23]

Corollary 1  Two generator matrices \( \Gamma, \Gamma' \in R^{k \times n} \) generate linearly \( d \)-isometric codes if and only if there is a group element

\[
(A, (\varphi; \pi)) \in (GL_{\lambda}(R)/N_{\lambda}(R)) \times (R^n/S_n)
\]

with \((A, \varphi; \pi)\Gamma := A ((\varphi; \pi)\Gamma) = \Gamma'.\)

The period of a column vector \( v^T \in R^k \) is invariant under the action of \( A \in GL_{\lambda}(R) \) and the multiplication of units from the right. Therefore, the mapping

\[
\varphi\text{period} \left( (GL_{\lambda}(R)/N_{\lambda}(R) \times R^n) \Gamma \right) := \left( \text{period}(\Gamma_{s,0}), \ldots, \text{period}(\Gamma_{s,n-1}) \right)
\]

defines an \( S_n \)-homomorphism. With the help of the Homomorphism Principle, we can suppose that the sequence \( \mu = (\mu_0, \ldots, \mu_{n-1}) \) of periods is decreasingly ordered and that the action is restricted to the stabilizer \( S_\mu \) of the sequence.

In general, the action by the column multiplications from the right is also not faithful on generator matrices whose column periods are equal to \( \mu = (\mu_0, \ldots, \mu_{n-1}) \). The kernel of this action is equal to \( R^\mu := \prod_{i=0}^{n-1} (1 + \text{Rad}(R)^{\mu_i}) \). Hence we may restrict the inner group action to

\[
G^{(\lambda, \mu)} := (GL_{\lambda}(R)/N_{\lambda}(R)) \times (R^n/R^\mu).
\]

4 Semicanonical representatives

As we have seen in Section 2, the efficient computation of an \( F \)-semicanonical representative of a generator matrix is a very crucial task during the backtracking algorithm for the computation of a canonical form of a linear code.

We will give some of our results and a short description of the algorithm for the computation of canonical forms under the action of \( G^{(\lambda, \mu)} \). By restricting the sequence \( F \) in the backtracking part to suitable subsequences of \( \text{Fixed}(\Omega) \), we are able to ensure that the matrix \( \Gamma' \) which has to be canonized is systematic. Hence this will be a general assumption in this section.

Again we define a generator matrix to be \( F \)-semicanonical if its projection \( \Gamma_{s,F} \) onto the columns defined by \( F \) is minimal in the orbit \( G^{(\lambda, \mu)} \Gamma \).
Proposition 3 Let $\Pi_{R^n}$ be the projection of the elements in $\text{Stab}_{G_{\lambda, \mu}}(\Gamma)$ onto their $R^n/R^H$ component. For $\varphi \in \Pi_{R^n}(\text{Stab}_{G_{\lambda, \mu}}(\Gamma))$ there is a uniquely defined element $A_{\varphi} \in \text{GL}_2(R)/N_2(R)$ such that $(A_{\varphi}, \varphi) \in \text{Stab}_{G_{\lambda, \mu}}(\Gamma)$. The mapping $\varphi \mapsto A_{\varphi}$ defines a group homomorphism. In particular, if $R$ is a commutative ring, then we know that $\text{Stab}_{G_{\lambda, \mu}}(\Gamma)$ is a commutative group. Furthermore, the matrix $A_{\varphi}$ is upper triangular (up to elements in $N_2(R)$).

Proposition 4 There is a subnormal series of $\text{Stab}_{G_{\lambda, \mu}}(\Gamma)$ where each factor is cyclic and whose length is bounded by $k + r \sum_{i \in |j|} (m - \lambda_i)$.

In the following we want to briefly summarize how we compute the canonical form $\text{CF}_{G_{\lambda, \mu}}(\Gamma)$ of a systematic generator matrix $\Gamma$ supposing that $\Gamma$ is already $(0, \ldots, n-2)$-semicanonical.

The algorithm is an inductive procedure depending on pairs $(i, x) \in [k] \times [m + 1]$. It starts at $(k - 1, 0)$ and computes its information at state $(i, x)$ from $(i, x - 1)$ if $x > 0$ or from $(i + 1, m)$ otherwise. The induction hypothesis is the following:

– The stabilizer

$$G_{\Gamma,(i,x)}^{(\lambda,\mu)} := \left\{ (\varphi, \psi) \in G_{\lambda, \mu} \mid \begin{array}{l}
(\psi, \varphi[n-1]) = (\Gamma_s[n-1]) \\
(\varphi \Gamma_{s, n-1}^{-1})_j = \Gamma_{j, n-1}, \forall j > i \\
(\varphi \Gamma_{s, n-1}^{-1})_i = \Gamma_{j, n-1} \in \text{Rad}(R)^x
\end{array} \right\}$$

has a subnormal series with cyclic factors and we let $E = (E_0, \ldots, E_{x-1})$ denote a corresponding sequence of generators, i.e. $\langle \{ \} \rangle < \langle E_{[y]} \rangle < \langle E_{[y+1]} \rangle < \ldots < \langle E_{[x]} \rangle$.

– For $k - 1 \geq j > i \geq 0$ we suppose that $\Gamma_{j, n-1}$ is minimal under the action of $G_{\Gamma,(j,0)}^{(\lambda,\mu)}$, i.e. $\Gamma_{j, n-1} = \min_{A \in G_{\Gamma,(j,0)}^{(\lambda,\mu)}} (A \Gamma)_{j, n-1}$.

– For $0 \leq y < x$ we suppose that $\text{coeff}^{(y)}(\Gamma_{s, n-1})$ is minimal under the action of $G_{\Gamma,(i,y)}^{(\lambda,\mu)}$, i.e. $\text{coeff}^{(y)}(\Gamma_{s, n-1}) = \min_{A \in G_{\Gamma,(i,y)}^{(\lambda,\mu)}} \text{coeff}^{(y)}((A \Gamma)_{s, n-1})$.

In this situation we can define the group homomorphism

$$\Phi_{(i,x, \Gamma)} : G_{\Gamma,(i,x)}^{(\lambda,\mu)} \rightarrow (F^*_q)^x \times (F^*_q)^x$$

$$(\varphi, \psi) \mapsto \left( \text{coeff}^{(y)}( (\varphi \varphi[n-1] - \Gamma_{s, n-1}) \cdot \text{coeff}^{(y)}(\varphi[n-1]^{-1}) \right)$$

where the multiplication on the right is defined by

$$(\alpha_0, \beta_0)(\alpha_1, \beta_1) := (\alpha_0 + \beta_0 \alpha_1, \beta_0 \beta_1)$$

The first component $\Phi_{1}^{(i,x, \Gamma)}((\alpha, \varphi))$ of this homomorphism returns the change that a group element $(\alpha, \varphi)$ will imply when applied to $\Gamma$. The generating set $E$ of $G_{\Gamma,(i,x)}^{(\lambda,\mu)}$ can be chosen in such a way that there is at most one generator $E_i$ with $\Phi_{1}^{(i,x, \Gamma)}(E_i) \neq 1$. The other generators define an $F_q$-subspace $H$ of $F_q$. 
Based on this observation it is easy to prove that the image $\Phi_{(i,x)}(\Gamma)(G(\lambda, \mu)(i,x))_0$ is a disjoint union of cosets of $H$ and therefore we can minimize $\text{coeff}^{(i)}_{(i,x)}(\Gamma_{i,n-1})$ via methods from linear algebra over $\mathbb{F}_p$. Using similar arguments we are furthermore able to update $E$ to be a generating set of $G^{(\lambda, \mu)}(i,x+1)$ which fulfills all the assumption we have made on it in our inductive process.

Altogether this shows that we are able to give a straightforward, efficient minimization algorithm for the action of $G^{(\lambda, \mu)}(i,x)$. This enables us to apply the partition refinement approach developed in Section 2 for linear codes over finite fields for the canonization of linear codes over finite chain rings as well.

The algorithm will be implemented in the computer algebra system Sage [9] and will be publicly available as an optional package soon.

References

When is $x^{-1} + L(x)$ a permutation in odd characteristic?

Faruk Göloğlu · Gary McGuire

Abstract Let $L(x)$ be a linearized polynomial on $\mathbb{F}_q$. Li and Wang proved that $x^{-1} + L(x)$ is a permutation if and only if $L(x) = 0$, in characteristic 2 and for $q \geq 32$. In this paper, we extend this result to odd characteristic, i.e., we prove that in odd characteristic, $x^{-1} + L(x)$ is a permutation if and only if

(i) $L(x) = 0$, or
(ii) $q = 3$ and $L(x) = x$, or
(iii) $q = 9$ and $L(x) = \omega^2x^3$ or $L(x) = \omega^6x^3$, where $\omega$ generates $\mathbb{F}_9^\ast$.

The proof uses Gauss sums and Kloosterman sums. We also prove a theorem which develops a well-known result of Carlitz. We use this theorem for the proof of our main result, but it may be of independent interest.

Keywords permutation polynomial · linearized polynomial · Kloosterman sum · Gauss sum

1 Introduction

Let $p$ be an odd prime. Let $\mathbb{F}_q$ be the finite field with $q = p^m$ elements and let $\text{Tr}$ denote the absolute trace map $\mathbb{F}_q \to \mathbb{F}_p$ defined by

$$\text{Tr}(x) = x + x^p + \cdots + x^{p^{m-1}}.$$ 

Recall that any function $\mathbb{F}_q \to \mathbb{F}_q$ can be expressed uniquely as a polynomial function (with coefficients in $\mathbb{F}_q$) of degree less than $q$. A polynomial function

Research supported by the Claude Shannon Institute, Science Foundation Ireland Grant 06/M1/006.
is called a permutation polynomial (PP) if it is a bijective function $F_q \rightarrow F_q$. Constructing PPs from simple functions is a topic of much interest. If $f(x)$ is a PP there have been many papers on PPs of the form $f(x) + ax$, and more generally $f(x) + L(x)$, where $L(x) = \sum a_i x^{p^i}$ is a linearized polynomial [2,11, 8,9,12,3,4]. In this paper we will give if and only if conditions for $x^{-1} + L(x)$ to be a PP. This has been conjectured [8] and proved in characteristic 2 in [9].

We denote the affine hyperplanes of $F_q$, viewed as an $F_p$-vector space, by

$$H^c_\alpha = \{ x \in F_q : \text{Tr}(\alpha x) = c \}$$

where $\alpha \in F_q^* = F_q \setminus \{0\}$ and $c \in F_p$. We note that, for any fixed $\alpha$, $F_q = \cup_{c \in F_p} H^c_\alpha$.

Define the set of (nonzero) squares and nonsquares as

$$\text{Sq} = \{ x^2 : x \in F_q^* \},$$
$$\text{NSq} = F_q^* \setminus \text{Sq}.$$

Let $\eta$ be the quadratic character of $F_q$, defined by

$$\eta(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x \in \text{Sq}, \\ -1 & \text{if } x \in \text{NSq}. \end{cases}$$

Let $\zeta$ be a primitive complex $p$-th root of unity. The additive characters of $F_q$ may be written

$$\chi_\alpha(x) = \zeta^{\text{Tr}(\alpha x)},$$

one character for each $\alpha \in F_q$. The identity (or principal) character is $\chi_0$, and $\chi_1$ is sometimes called the canonical additive character of $F_q$.

Of particular interest to us in this paper are the characters $\chi_\alpha$ where $\alpha \in F_p$. For $c \in F_p$ we will write

$$\chi_c(x) = \zeta^{\text{Tr}(cx)} = \zeta^{c \text{Tr}(x)}.$$

We denote the quadratic Gauss sum by

$$G(\eta, \chi_c) = \sum_{x \in F_q^*} \eta(x) \chi_c(x).$$

The celebrated formula for quadratic Gauss sums is as follows [10, Theorem 5.15]:

$$G(\eta, \chi_c) = \begin{cases} \eta(c)(-1)^{m-1} \sqrt{q} & \text{if } p \equiv 1 \pmod{4}, \\ \eta(c)(-1)^{m-1} i^{m} \sqrt{q} & \text{if } p \equiv 3 \pmod{4}. \end{cases}$$

(1)

For the trivial character we note that $G(\eta, \chi_0) = 0$. 

2 A preliminary lemma

In the next lemma we will compute the hyperplane sums

\[ S^{(c)}_{\alpha} = \sum_{x \in H^{(c)}_{\alpha}} \eta(x) \]

for any \( \alpha \in \mathbb{F}_q^* \) and \( c \in \mathbb{F}_p \). This result may not be new, but we do not know a reference. The result in characteristic 3 was proved in [6].

**Lemma 1** If \( m \) is even then

\[
S^{(0)}_{\alpha} = \begin{cases} 
-\eta(\alpha)(p-1)p^{(m-2)/2} & \text{if } p \equiv 1 \pmod{4}, \\
\eta(\alpha)(p-1)(-p)^{(m-2)/2} & \text{if } p \equiv 3 \pmod{4},
\end{cases}
\]

and, for \( c \in \mathbb{F}_p^* \),

\[
S^{(c)}_{\alpha} = \begin{cases} 
\eta(\alpha)p^{(m-2)/2} & \text{if } p \equiv 1 \pmod{4}, \\
-\eta(\alpha)(-p)^{(m-2)/2} & \text{if } p \equiv 3 \pmod{4}.
\end{cases}
\]

If \( m \) is odd then

\[ S^{(0)}_{\alpha} = 0, \]

and, for \( c \in \mathbb{F}_p^* \),

\[
S^{(c)}_{\alpha} = \begin{cases} 
\eta(\alpha)c^{(m-2)/2} & \text{if } p \equiv 1 \pmod{4}, \\
\eta(\alpha)c^{(m-2)/2} & \text{if } p \equiv 3 \pmod{4}.
\end{cases}
\]

**Proof** Let \( \alpha \in \mathbb{F}_q^* \) and let \( c \in \mathbb{F}_p \). We have

\[
G(\eta, \chi_c) = \sum_{x \in \mathbb{F}_q^*} \eta(x)\chi_c(x) \\
= \sum_{x \in \mathbb{F}_q^*} \eta(\alpha x)\chi_c(\alpha x) \\
= \eta(\alpha) \sum_{d \in \mathbb{F}_p} \sum_{x \in H^{(c)}_{\alpha}} \eta(x)\zeta^{c \text{Tr}(\alpha x)} \\
= \eta(\alpha) \sum_{d \in \mathbb{F}_p} \zeta^{cd} S^{(d)}_{\alpha}.
\]

Therefore

\[
\sum_{c \in \mathbb{F}_p} G(\eta, \chi_c) = \eta(\alpha) \sum_{c \in \mathbb{F}_p} \left( \sum_{d \in \mathbb{F}_p} \zeta^{cd} S^{(d)}_{\alpha} \right) \\
= \eta(\alpha) \sum_{d \in \mathbb{F}_p} S^{(d)}_{\alpha} \left( \sum_{c \in \mathbb{F}_p} \zeta^{cd} \right) \\
= p \eta(\alpha) S^{(0)}_{\alpha}. \tag{2}
\]
This is true for any fixed $\alpha \in \mathbb{F}_q^\ast$.

Next we derive a different expression. Note that $x^2 - c$ has a solution in $\mathbb{F}_{p^m}$ for all $c \in \mathbb{F}_p$ if and only if $m$ is even. Note also that if $m$ is even then $S^{(c)}_\alpha = S^{(d)}_\alpha$ for all $c, d \in \mathbb{F}_p^\ast$, and if $m$ is odd then $S^{(c)}_\alpha = \eta(d/c)S^{(d)}_\alpha$. These observations together with (1) prove that, if $m$ is odd

$$\sum_{c \in \mathbb{F}_p} G(\eta, \chi_c) = 0,$$

(3)

and if $m$ is even

$$\sum_{c \in \mathbb{F}_p} G(\eta, \chi_c) = (p - 1)G(\eta, \chi_1).$$

(4)

Hence if $m$ is even by combining (1), (2) and (4)

$$S^{(0)}_\alpha = \begin{cases} -\eta(\alpha)(p - 1)p^{(m - 2)/2} & \text{if } p \equiv 1 \pmod{4}, \\
\eta(\alpha)(p - 1)(-p)^{(m - 2)/2} & \text{if } p \equiv 3 \pmod{4}, \end{cases}$$

and for $c \in \mathbb{F}_p^\ast$, using the formula for $S^{(0)}_\alpha$ we have

$$S^{(c)}_\alpha = \begin{cases} \eta(\alpha)p^{(m - 2)/2} & \text{if } p \equiv 1 \pmod{4}, \\
-\eta(\alpha)(-p)^{(m - 2)/2} & \text{if } p \equiv 3 \pmod{4}, \end{cases}$$

If $m$ is odd then by expanding the Gauss sum we get

$$G(\eta, \chi_1) = \eta(\alpha)S^{(1)}_\alpha \left( \sum_{c \in \mathbb{F}_p} \eta(c)\zeta^c \right)$$

and by using (1) on $\mathbb{F}_p$ we have

$$G(\eta, \chi_1) = \begin{cases} \eta(\alpha)\sqrt{p}S^{(1)}_\alpha & \text{if } p \equiv 1 \pmod{4}, \\
\eta(\alpha)i\sqrt{p}S^{(1)}_\alpha & \text{if } p \equiv 3 \pmod{4}. \end{cases}$$

Therefore by (2) and (3) we have

$$S^{(0)}_\alpha = 0,$$

and, for $c \in \mathbb{F}_p^\ast$, using (1) once more, we have

$$S^{(c)}_\alpha = \begin{cases} \eta(\alpha)c^{m-1} & \text{if } p \equiv 1 \pmod{4}, \\
\eta(\alpha)c(-p)^{m-1} & \text{if } p \equiv 3 \pmod{4}. \end{cases}$$

$\Box$
3 A generalization of a theorem of Carlitz

The following theorem is a well-known theorem due to Carlitz.

**Theorem 1 (Carlitz [1])** Let \( f(x) \) be a polynomial over \( \mathbb{F}_q[x] \) such that \( f(0) = 0, f(1) = 1, \) and

\[
\eta(f(a) - f(b)) = \eta(a - b)
\]

(5)

for all \( a,b \in \mathbb{F}_q \). Then \( f(x) = x^q \) for some \( 0 \leq d < m \).

We remark that if \( f(x) \) is not a permutation polynomial then condition (5) cannot be satisfied, because there will exist \( a \neq b \) (and so \( \eta(a-b) \neq 0 \)) for which \( f(a) - f(b) = 0 \) (and so \( \eta(f(a) - f(b)) = 0 \)). We therefore modify condition (5) as follows:

\[
\eta(f(a) - f(b)) \eta(a - b) \in \{0, 1\},
\]

(6)

for all \( a,b \in \mathbb{F}_q \). If \( f \) is a PP then condition (5) is equivalent to condition (6), because assuming \( f \) is a PP, both \( \eta(a - b) \) and \( \eta(f(a) - f(b)) \) are only 0 when \( a = b \), and when both are nonzero their product is 1 if and only if they are equal.

Recall that \( L \) is a linearized polynomial in \( \mathbb{F}_q[x] \) if \( L(x) = a_0x + a_1x^q + a_2x^{q^2} + \cdots + a_{m-1}x^{q^{m-1}} \) with \( a_i \in \mathbb{F}_q \). If \( f = L \) is linearized then the condition (6) is equivalent to

\[
\eta(a)\eta(L(a)) \in \{0, 1\},
\]

(7)

for all \( a \in \mathbb{F}_q \). This in turn is equivalent to

\[
\eta(aL(a)) \in \{0, 1\},
\]

(8)

The following theorem can therefore be seen as a variation of Carlitz’ result. It is a generalization for linearized polynomials only.

**Theorem 2** If \( L(x) \) is a linearized polynomial then \( \text{Im}(xL(x)) \subseteq \mathbb{S} \cup \{0\} \) if and only if \( L(x) = 0 \) or \( L(x) = ax^{q^d} \) for some \( a \in \mathbb{S} \) and some \( 0 \leq d < m \).

**Proof** If \( L(x) = ax^{q^d} \) for some \( a \in \mathbb{S} \cup \{0\} \) and \( 0 \leq d < m \) then condition (8) is satisfied because \( xL(x) = ax^{q^{d+1}} \), and \( q^d + 1 \) is even.

Conversely, assume \( \text{Im}(xL(x)) \subseteq \mathbb{S} \cup \{0\} \). If \( L(x) \) is a PP then the result follows by Theorem 1, and if \( L(x) = 0 \) then the result is obvious. We will now assume that \( L(x) \) is nonzero and is not a PP, and we will derive a contradiction.

Let \( K \) be the kernel of \( L(x) \) and \( V \) be a subspace satisfying the direct sum \( K \oplus V = \mathbb{F}_q \). Let \( k = \dim(K) \), and since \( L(x) \) is nonzero and not a PP we have \( m > k > 0 \). Therefore \( \dim(V) = m - k \). It is clear that for any \( v \in V \)
and $x \in K$ we have $L(x + v) = L(v)$, i.e., $L$ is constant on any coset $K + v$. Therefore, for any $v \in V \setminus \{0\}$ and any $x \in K$ the hypothesis gives

$$1 = \eta((x + v)L(x + v)) = \eta(x + v) \eta(L(v))$$

and hence

$$\eta(x + v) = \eta(L(v)). \tag{9}$$

This means that $\eta$ is constant on cosets $K + v$ for nonzero $v$. Then

$$0 = \sum_{y \in \mathbb{F}_q} \eta(y) = \sum_{x \in K} \sum_{v \in V} \eta(x + v) = \sum_{x \in K} \eta(x) + \sum_{x \in K} \sum_{v \in V \setminus \{0\}} \eta(L(v))$$

$$= \sum_{x \in K} \eta(x) + p^k \sum_{v \in V \setminus \{0\}} \eta(L(v)).$$

Each of these sums is an integer, and because $\eta(0) = 0$ the first sum is at most $p^k - 1$ in absolute value. Therefore we must have both

$$\sum_{x \in K} \eta(x) = 0 \quad \text{and} \quad \sum_{v \in V \setminus \{0\}} \eta(L(v)) = 0.$$

Next, choose $\beta \in \mathbb{F}_q$ such that $K \subseteq H^{(0)}_\beta$ (this is possible since $k \neq m$). Choose a subspace $V_0$ of codimension 1 in $V$ such that $K \oplus V_0 = H^{(0)}_\beta$. Let $V_i$ ($0 \leq i \leq p - 1$) be the cosets of $V_0$ in $V$. Then, for any nonzero $i$, by using (9) we may write

$$S^{(i)}_\beta = \sum_{x \in K} \sum_{v \in V_i} \eta(x + v) = p^k \sum_{v \in V_i} \eta(L(v)),$$

which implies that $S^{(i)}_\beta$ is divisible by $p^k$. From Lemma 1 we know the exact power of $p$ that divides $S^{(i)}_\beta$, and it is $\left\lfloor \frac{m - 1}{2} \right\rfloor$. Therefore

$$k \leq \left\lfloor \frac{m - 1}{2} \right\rfloor. \tag{10}$$

On the other hand, choose $\gamma \in \mathbb{F}_q$ such that $V \subseteq H^{(0)}_\gamma$ (this is possible since $k \neq 0$). Choose a subspace $K_0$ of codimension 1 in $K$ such that $V \oplus K_0 = H^{(0)}_\gamma$. Let $K_i$ ($0 \leq i \leq p - 1$) be the cosets of $K_0$ in $K$. Then, for any nonzero $i$,

$$S^{(i)}_\gamma = \sum_{v \in V} \sum_{x \in K_i} \eta(x + v)$$

$$= \sum_{x \in K_i} \eta(x) + \sum_{x \in K_i} \sum_{v \in V \setminus \{0\}} \eta(x + v)$$

$$= \sum_{x \in K_i} \eta(x) + p^{k-1} \sum_{v \in V \setminus \{0\}} \eta(L(v))$$

$$= \sum_{x \in K_i} \eta(x).$$
This implies that $S_{y}^{(i)}$ is at most $p^{k-1}$ in absolute value. But from Lemma 1 we know that $S_{y}^{(i)}$ is nonzero and is divisible by $p^{|\frac{m-1}{2}|}$, so

$$k - 1 \geq \left\lceil \frac{m - 1}{2} \right\rceil$$

which contradicts (10).

\[ \square \]

4 Permutation behaviour of $x^{-1} + L(x)$

A Kloosterman sum is defined by

$$K(a) = \sum_{x \in \mathbb{F}_q} \zeta^{\text{Tr}(x^{-1} + ax)} = \sum_{x \in \mathbb{F}_q} \chi_1(x^{-1} + ax).$$

We denote the zero-set of Kloosterman sums by $\mathcal{KZ} = \{ a \in \mathbb{F}_q : K(a) = 0 \}$. If $p > 3$ then we have $\mathcal{KZ} = \{ 0 \}$ (see [7, Corollary 2]). We have the following characterization of ternary Kloosterman sums modulo 4 from [6,5].

**Theorem 3** Let $a \in \mathbb{F}_{3^m}$. Then

$$K(a) \equiv \begin{cases} 
0 & \text{mod 4 if } a = 0 \text{ or } a = b^2 \text{ with } \text{Tr}(b) = 1 \text{ and } -b \text{ is not a square}, \\
2m + 3 & \text{mod 4 if } a = t^2 - t^3 \text{ for some } t \in \mathbb{F}_q \setminus \{0,1\} \text{ and at least one of } t, 1 - t \text{ is a square}, \\
2 & \text{mod 4 if } a = b^2 \text{ with } \text{Tr}(b) = 1 \text{ and } -b \text{ is a square}, \\
2m + 1 & \text{mod 4 if } a = t^2 - t^3 \text{ for some } t \in \mathbb{F}_q \setminus \{0,1\} \text{ and none of } t, 1 - t \text{ is a square}.
\end{cases}$$

The following theorem is well-known (cf. [10, Theorem 7.7])

**Theorem 4** A polynomial $P(x)$ in $\mathbb{F}_q[x]$ is a permutation polynomial if and only if

$$\sum_{x \in \mathbb{F}_q} \chi(P(x)) = 0$$

for every nontrivial additive character $\chi$ of $\mathbb{F}_q$.

Let $L(x) = \sum_{i=0}^{m-1} a_i x^{p^i}$, and recall that the adjoint of $L$ is

$$L^*(x) = \sum_{i=0}^{m-1} a_i^{p^{m-i}} x^{p^{m-i}}.$$

The adjoint has the property that

$$\text{Tr}(xL(y)) = \text{Tr}(L^*(x)y).$$

In the following we will need the condition of Hermite [10, Theorem 7.4].
Theorem 5 (Hermite’s criterion) A polynomial \( f \in \mathbb{F}_{p^n}[x] \) is a permutation polynomial if and only if

1. \( f \) has exactly one root in \( \mathbb{F}_{p^n} \).
2. for each \( d \) with \( 1 \leq d \leq p^n - 2 \) and \( d \not\equiv 0 \pmod{p} \), the degree of \( f(x)^d \) (mod \( x^{p^n} - x \)) is less than \( p^n - 1 \).

The following is the main result of this section.

Theorem 6 If \( p \) is odd then \( x^{-1} + L(x) \) is a PP if and only if

(i) \( L(x) = 0 \), or
(ii) \( q = 3 \) and \( L(x) = x \), or
(iii) \( q = 9 \) and \( L(x) = \omega^2 x^3 \) or \( L(x) = \omega^5 x^3 \), where \( \omega \) generates \( \mathbb{F}_q^\ast \).

Proof Replacing \( x \) by \( ax \) shows

\[
\sum_{x \in \mathbb{F}_q} \chi_1 \left( \alpha (x^{-1} + L(x)) \right) = \sum_{x \in \mathbb{F}_q} \chi_1 \left( x^{-1} + \alpha L^\ast (\alpha) x \right).
\]

Recalling that the characters of \( \mathbb{F}_q \) are of the form \( x \mapsto \chi_1(\alpha x) \), it follows from Theorem 4 that \( x^{-1} + L(x) \) is a PP if and only if \( \sum_{x \in \mathbb{F}_q} \chi_1(x^{-1} + \alpha L^\ast(\alpha) x) = 0 \) for all \( \alpha \). In other words, \( x^{-1} + L(x) \) is a PP if and only if \( \text{Im}(xL^\ast(x)) \subseteq KZ \).

It is clear that \( L(x) = 0 \) implies \( x^{-1} + L(x) = x^{-1} \) is a PP.

If \( p > 3 \) then \( KZ = \{0\} \) by [7]. The hypothesis \( \text{Im}(xL^\ast(x)) = \{0\} \) certainly implies \( L(x) = 0 \). This completes the proof in the case \( p > 3 \).

Now let \( p = 3 \). We will show that

\[
\text{Im}(xL^\ast(x)) \subseteq KZ
\]

is not possible unless \( L(x) \) is one of (i), (ii) or (iii). By Theorem 3, \( KZ \subseteq Sq \cup \{0\} \). By Theorem 2, we have \( \text{Im}(xL^\ast(x)) \subseteq Sq \cup \{0\} \) if and only if \( L(x) = 0 \) or \( L(x) = ax^{p \cdot d'} \) for some \( a \in Sq \) and \( 0 \leq d < m \) (note that if \( L(x) = ax^{p \cdot d'} \) then \( L^\ast(x) = a^{p \cdot m - d} x^{p \cdot m - d} \)).

Assume \( m > d > 0 \) and \( d \neq m - d \). We use the Hermite Criterion, and we will show that there exists a \( k \) such that the degree of \( (x^{-1} + ax^{p \cdot d})^k \) is \( p^m - 1 \). Choose \( k = p^{m-d} + 1 \). We have

\[
(x^{p^m-2} + ax^{p^m})^{p^{m-d}+1} = x^{(p^m-2)(p^{m-d}+1)} + a^{p^{m-d}+1} x^{p^m(p^{m-d}+1)}
+ a^{p^{m-d}} x^{(p^m-2)p^{m-d}+d} + ax(p^m-2)(p^{m-d}+d) + p^d.
\]

Now consider the following congruences

\[
(p^m - 2)(p^{m-d} + 1) \equiv 0 \pmod{p^m - 1},
\]
\[
p^d(p^{m-d} + 1) \equiv 0 \pmod{p^m - 1},
\]
\[
(p^m - 2) + p^d(p^{m-d}) \equiv 0 \pmod{p^m - 1},
\]
\[
(p^m - 2)(p^{m-d}) + p^d \equiv 0 \pmod{p^m - 1}.
\]
If any one of these is not satisfied, we are done. These become
\[
(p^m - 1)(p^{m-d} + 1) - (p^{m-d} + 1) \equiv 0 \pmod{p^m - 1},
\]
\[
1 + p^d \equiv 0 \pmod{p^m - 1},
\]
\[
(p^m - 1) + 1 - 1 \equiv 0 \pmod{p^m - 1},
\]
\[
(p^m - 1)(p^{m-d}) - p^{m-d} + p^d \equiv 0 \pmod{p^m - 1}.
\] (11)

If \( d \neq m - d \mod m \), only Eq. (11) is satisfied. Therefore the polynomial \( x^{-1} + ax^{p^d} \) is not a PP and we are done unless \( d = m - d \) or \( d = 0 \).

If \( d = 0 \) then \( f(x) = x^{-1} + ax \) with \( a \in \mathbb{F}_q^* \) has the property that \( f(x) = f(1/ax) \) so \( f(x) \) is a PP if and only if \( q = 3 \) and \( a = 1 \).

The remaining case is when \( d = m - d \mod m \), so \( d = m/2 \). Let \( f(x) = x^{-1} + ax^{p^d} \) with \( a \in \mathbb{F}_q^* \). Then if \( f \) is a PP then \( \text{Im}(xL^*(x)) = \text{Im}(x^{p^d+1}a^{p^d}) \subseteq KZ \). Note that \( \text{Im}(x^{p^d+1}a^{p^d}) = a^{p^d} \mathbb{F}_{p^m} \) and that every nonzero element of \( \mathbb{F}_{p^m} \) is a square. Since \( \mathbb{F}_{p^m} \) is the quadratic extension over \( \mathbb{F}_{p^d} \), we must have \( a^{p^d} = b^2 \) for some nonsquare \( b \in \mathbb{F}_{p^m} \), which satisfies \( \text{Tr}(b) \neq 0 \) by Theorem 3.

Now consider the square roots of \( \text{Im}(xL^*(x)) \). We have \( b \mathbb{F}_{p^d} \subseteq \text{Im}(\sqrt{xL^*(x)}) \). Consider now the set \( T_b = \{ \text{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_{p^d}}(bc) : c \in \mathbb{F}_{p^d}^* \} \). We have either \( T_b = \mathbb{F}_{p^d}^* \) or \( T_b = \{ 0 \} \). Therefore there exists \( c \in \mathbb{F}_{p^d}^* \) such that \( \text{Tr}(bc) = 0 \) if \( d > 1 \). Since \( (bc)^2 \in \text{Im}(xL^*(x)) \) we have proved that if \( f \) is a PP then \( d = 1 \) (and therefore \( m = 2 \)). Finally, if \( d = 1 \) then the set \( \omega^2 \mathbb{F}_{p^d} \subseteq KZ = \{ 0, \omega^2, \omega^6 \} \) and \( \omega^2 \mathbb{F}_{p^d} \subseteq KZ \), showing \( x^{-1} + \omega^2 x^1 \) and \( x^{-1} + \omega^2 x^3 \) are both permutations. Indeed, it is a simple matter to check none of the square roots \( \omega, \omega^5, \omega^3, \omega^7 \) of the elements of \( \text{Im}(xL^*(x)) \) have 0 trace, since there are only three elements with 0 trace in \( \mathbb{F}_9 \) but \( \text{Tr}(\omega) = \text{Tr}(\omega^3) = - \text{Tr}(\omega^5) = - \text{Tr}(\omega^7) \). Noting that \( \omega^2 \) and \( \omega^6 \) are the only squares of \( \mathbb{F}_9 \) which are not 4-th powers, we complete the proof.

\[ \Box \]

Remark 1 For \( p > 3 \) similar arguments show that the polynomial \( L_1(x^{-1}) + L_2(x) \) is a PP if and only if \( L_2(x) = 0 \) (resp. \( L_1(x) = 0 \)) and \( L_1(x) \) is a PP (resp. \( L_2(x) \) is a PP).

References

2. Charpin, P., Kyureghyan, G.: When does \( G(x)+\gamma \text{Tr}(H(x)) \) permute \( \mathbb{F}_{p^n} \)? Finite Fields Appl. 15(5), 615–632 (2009)
On the exact number of solutions of certain linearized equations

Ferruh Özbudak · Zülfükar Saygı

1 Introduction

In this note we have revisited some of the results of Trachtenberg [9], which are directly related with the number of solutions of some special linearized polynomials over finite fields. In some cases we give improvements. Also, we give some results on the exact number of solutions of certain linearized equations depending on the coefficients of that equation.

Keywords Linearized equations · Finite fields
Lemma 1 Let $B \subseteq \mathbb{F}_{p^n}$ be a non-empty set. If $B$ is linearly independent over $\mathbb{F}_{q_1}$, then $B$ is also linearly independent over $\mathbb{F}_q$.

Proof Here, we give a sketch of the proof only. The proof follows from the observation that $n_1 = [\mathbb{F}_{q_2} : \mathbb{F}_q]$ is equal to $[\mathbb{F}_{p^n} : \mathbb{F}_{q_1}]$ and some arguments from field extensions. \hfill \Box

Here we note that this lemma is a stronger version of [9, Lemma 4]. Using this lemma one can easily decide the possible values for number of solutions of certain linearized equations. But it is not easy to find the exact number of solutions of that equation. In the following section we will give some results concerning this problem. In many cases we can easily find the exact number of solutions of linearized equations depending on the coefficients of that equation.

2 Main Results

Recall that $n_1 = n/e$. Throughout this section we assume that $n_1 \geq 2$ and we will use the notation Norm for the relative norm map from $\mathbb{F}_{p^n}$ to $\mathbb{F}_{q_1}$ (that is, $\text{Norm}(x) = x^{q_1^{n_1} - 1}$ for any $x \in \mathbb{F}_{p^n}$). We start with a useful proposition.

Proposition 1 Let $\alpha \in \mathbb{F}_{p^n} \setminus \{0\}$ and $N(\alpha)$ denote the number of $z \in \mathbb{F}_{p^n}$ such that $z^{q_1} - \alpha z = 0$.

Let $\psi_\alpha$ be the map on $\mathbb{F}_{p^n}$ given by $\psi_\alpha : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ $x \mapsto x^{q_1} - \alpha x$.

Then we have $N(\alpha) = \begin{cases} 1, & \text{if } \text{Norm}(\alpha) \neq 1, \\ q_1, & \text{if } \text{Norm}(\alpha) = 1. \end{cases}$
Let $A_\alpha(T) \in \mathbb{F}_{p^n}[T]$ be the $\mathbb{F}_{q_1}$-linearized polynomial given by

$$A_\alpha(T) = T^{q_1^{n-1}} + a_1 T^{q_1^{n-2}} + a_2 T^{q_1^{n-3}} + \cdots + a_n T.$$  \hfill (1)

If $\text{Norm}(\alpha) = 1$, then we also have the followings:

1. $\text{Ker}\psi_\alpha$ is the roots of the polynomial $T^{q_1} - \alpha T$ over $\mathbb{F}_{p^n}$. This polynomial is separable and splits over $\mathbb{F}_{p^n}$.
2. $\text{Im}\psi_\alpha$ is the roots of the polynomial $A_\alpha(T)$. This polynomial is separable and splits over $\mathbb{F}_{p^n}$.

Using Proposition 1 we obtain the following result.

**Theorem 1** Let $\alpha, \beta$ be nonzero elements of $\mathbb{F}_{p^n}$. Let $N(\alpha, \beta)$ denote the number of $z \in \mathbb{F}_{p^n}$ such that

$$(z^q - \alpha z) \circ (z^q - \beta z) = z^q - (\alpha + \beta^q) z^q + \alpha \beta z = 0.$$ 

Let $C_{\alpha, \beta}$ denote the constant in $\mathbb{F}_{p^n}$ defined as

$$C_{\alpha, \beta} = \frac{1}{\beta} + \frac{a_1}{\beta^{q_1+1}} + \frac{\alpha}{\beta^{q_1+1} q_1 + q_1 + 1} + \cdots + \frac{\alpha^{q_1^{n-3} + \cdots + q_1^{n-1} + 1}}{\beta^{q_1^{n-2} + \cdots + q_1 + 1}} + \frac{\alpha^{q_1^{n-2} + \cdots + q_1 + 1}}{\beta^{q_1^{n-1} + q_1 + 1}}.$$ 

Then $N(\alpha, \beta) \in \{1, q_1, q_1^2\}$. Moreover we have the followings:

1. $N(\alpha, \beta) = 1$ if and only if $\text{Norm}(\alpha) \neq 1$ and $\text{Norm}(\beta) \neq 1$.
2. $N(\alpha, \beta) = q_1$ if and only if one of the followings hold:
   (a) $\text{Norm}(\alpha) = 1$ and $\text{Norm}(\beta) \neq 1$.
   (b) $\text{Norm}(\alpha) \neq 1$ and $\text{Norm}(\beta) = 1$.
   (c) $\text{Norm}(\alpha) = \text{Norm}(\beta) = 1$ and $C_{\alpha, \beta} \neq 0$.
3. $N(\alpha, \beta) = q_1^2$ if and only if $\text{Norm}(\alpha) = \text{Norm}(\beta) = 1$ and $C_{\alpha, \beta} = 0$.

**Proof** Here, we give a sketch of the proof only. The proof uses Proposition 1 and the observation that $(T^{q_1} - \alpha T) \mid A_\beta(T)$ if and only if $C_{\alpha, \beta} = 0$, where $A_\beta(T)$ is the $\mathbb{F}_{q_1}$-linearized polynomial in $\mathbb{F}_{p^n}[T]$ defined in (1). \hfill $\square$

The following result is well known if $k \mid n$. Here we give a slight extension, including the case $k \nmid n$ as well. We will use the following proposition later.

**Proposition 2** Let $m \geq 2$ be an integer. Let

$$A(T) = T^{q^m} + A_{m-1} T^{q^{m-1}} + \cdots + A_1 T^q + A_0 T \in \mathbb{F}_{p^n}[T]$$

be an $\mathbb{F}_q$-linearized polynomial with $A_0 \neq 0$. If there exists $\eta \in \mathbb{F}_{p^n} \setminus \{0\}$ such that $A(\eta) = 0$, then there exists $\beta \in \mathbb{F}_{p^n} \setminus \{0\}$ and $\mathbb{F}_q$-linearized monic and separable polynomial $B(T) \in \mathbb{F}_{p^n}[T]$ such that

$$A(T) = B(T) \circ (T^q - \beta T).$$
Proof Note that $F_q \subseteq F_{q^2}$. Considering $A(T) \in F_{q^2}[T]$ we obtain that there exists $B(T) \in F_{p^n}[T]$ such that

$$A(T) = B(T) \circ \left( \prod_{c \in F_q} (T - c\eta) \right).$$

(2)

Here

$$\prod_{c \in F_q} (T - c\eta) = T^q - \beta T$$

(3)

with $\beta \in F_q$ and $B(T) \in F_{q^2}[T]$. It remains to prove that $\beta \in F_{p^n}$. Indeed if $T^q - \beta T \in F_{p^n}[T]$, then using (2) we conclude that $B(T) \in F_{p^n}[T]$, which completes the proof.

Now we prove that $\beta \in F_{p^n}$. Comparing the coefficients of degree 1 terms in both sides of (3) we obtain that

$$\prod_{c \in F_q \setminus \{0\}} (-c\eta) = -\beta \quad \text{and hence} \quad \beta = -\eta^{q-1} \prod_{c \in F_q \setminus \{0\}} = \eta^{q-1}. $$

This shows that $\beta \in F_{p^n} \setminus \{0\}$ as $\eta \in F_{p^n} \setminus \{0\}$.

Let $a, b \in F_{p^n} \setminus \{0\}$. Let $N$ denote the number of $z \in F_{p^n}$ such that

$$z^{q^2} + az^q + bz = 0. $$

The main problem of this section is to compute $N$ explicitly. This problem is now reduced to a “factorization” problem in the following sense:

- If there exist $\alpha, \beta \in F_{p^n} \setminus \{0\}$ such that

$$z^{q^2} + az^q + bz = (z^q - \alpha z) \circ (z^q - \beta z),$$

(4)

then $N$ is computed explicitly using Theorem 1 as $N = N(\alpha, \beta)$.

- If there is no $\alpha, \beta \in F_{p^n} \setminus \{0\}$ such that (4) holds, then $N = 1$ by Proposition 2.

The following is a numerical example such that there is no $\alpha, \beta \in F_{p^n} \setminus \{0\}$ satisfying (4).

Example 1 Let $p = 3$, $n = 3$, $k = 1$ and $\gamma$ be a primitive element in $F_{3^3}$, such that $\gamma^3 + 2\gamma + 1 = 0$. Then by computer search we see that

$$z^9 + z^3 + z$$

can not be written of the form $(z^3 - \alpha z) \circ (z^3 - \beta z)$ for all $\alpha, \beta \in F_{3^3} \setminus \{0\}$.
We find it interesting to present a connection of the factorization problem above with a result of Bluher [1]. Now we want to find $\alpha, \beta \in \mathbb{F}_{p^n} \setminus \{0\}$ such that
\[
z^q^2 + az^q + bz = (z^q - \alpha z) \circ (z^q - \beta z) = z^{q^2} - (\alpha + \beta^q) z^q + \alpha \beta z,
\]
which means that
\[
a = \alpha + \beta^q \text{ and } b = \alpha \beta.
\]
Then by substituting $\alpha = b/\beta$ in the first equality we get
\[
a = \frac{b}{\beta} + \beta^q
\]
that is, $\beta$ is a solution of the equation
\[
0 = x^{q+1} - ax + b \in \mathbb{F}_{p^n}[x]
\]
which is studied in more detail in [1].

Now we note that, using Lemma 1 Trachtenberg obtained the following result [9, Proposition 1].

**Proposition 3** Let $\gamma$ be a nonzero element of $\mathbb{F}_{p^n}$ where $p$ is prime and $n$ is odd. Then the equation
\[
z^{p^m} - (2\gamma)p^{2m}z^{p^{2m}} + z = 0
\]
has exactly $1$, $p^e$, or $p^{2e}$ roots in $\mathbb{F}_{p^n}$, where $e = \gcd(m,n)$.

Here we remark that using Theorem 1 and Proposition 2 it possible to find the exact number of roots of (5) depending on $\gamma$. So this gives an improvement. We note that $k = 2m$ in our notation.

Furthermore, using the above techniques it is possible to find the exact number of solutions of the following linearized equations depending on the coefficients of that equation.
\[
z^q^3 + az^q^2 + bz^q + cz = 0.
\]
Similarly, the problem of finding the exact number of solutions of (6) can be reduced to a “factorization” problem as above. Here we note that using Lemma 1 one can say that the number of solutions of (6) is in the set $\{1, q_1, q_1^2, q_1^3\}$. But in many cases depending on the coefficients of the equations, the number of solutions of (6) will not take all the values in the set $\{1, q_1, q_1^2, q_1^3\}$. For example, we can give the following proposition. It is given in [9] and it is noted in [9] that the proof of the proposition is suggested by L. Welch.

**Proposition 4** Let $\gamma$ be a nonzero element of $\mathbb{F}_{p^n}$ where $p$ is an odd prime and $n$ is odd. Then the equation
\[
z^{p^6m} - \gamma p^{3m}z^{p^{3m}} - \gamma p^{2m}z^{p^{2m}} + z = 0
\]
has exactly $1$, $p^e$, or $p^{2e}$ roots in $\mathbb{F}_{p^n}$, where $e = \gcd(m,n)$. 
In our notation by setting $k = 2m$, we see that the equation (7) is of the form
\[ 0 = z^q + b^{(k/2)} z^q + b z^q + z \in \mathbb{F}_{p^n}[z]. \]
Note that, if the equation (7) has a nonzero root then it is shown in [9] that
\[ z^q + b^{(k/2)} z^q + b z^q + z = (z^q - \alpha_1 z) \circ (z^q - \alpha_2 z) \circ (z^q - \alpha_3 z) \]
for some $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}_{p^n}$. Using this observation it is proved in [9] that the
equation (7) can not have $q^3$ solutions in $\mathbb{F}_{p^n}$. Furthermore, using similar tech-
niques as in Theorem 1 it is possible to make further improvements depending
on the values of $\alpha_1$, $\alpha_2$ and $\alpha_3$.

**Acknowledgments.** The authors were partially supported by TÜBİTAK under Grant No. TBAG–109T672. The work of Z. Saygı was also supported by TÜBİTAK under Grant No. TBAG–109T344.

**References**

2. S. -T. Choi, J. -S. No, H. Chung, On the cross-correlation of a ternary m-sequence of
period $3^{4k+2} - 1$ and its decimated sequence by $\frac{(3^{2k+1}+1)^2}{8}$, *ISIT 2010*, 1268–1271, (2010).
p-ary m-sequence and its decimated sequence by $\frac{p+1}{p+1} + \frac{p-1}{2}$, *ISIT 2011*, 683–687, (2011).
m-Sequence of Period $p^{2m} - 1$ and Its Decimated Sequences by $(p^{2m} + 1)^2/2(p + 1)$,
5. T. Helleseth, G. Gong, New nonbinary sequences with ideal two-level autocorrelation,
6. T. Helleseth, G. Gong, H. Hu, A Three-Valued Walsh Transform From Decimations
of Helleseth-Gong Sequences, *IEEE Transactions on Information Theory*, 58(2), 1158–
1162, (2012).
7. T. Helleseth, G. Gong, H. Hu, C. Li, New Three-Valued Walsh Transforms from Deci-
8. J. Luo, T. Helleseth, A. Kholosha, Two nonbinary sequences with six-valued cross
Abstract In this paper, we concentrate on the very recently proposed Measurement Device Independent Quantum Key Distribution (MDI QKD) protocol by Lo, Curty and Qi (PRL, 2012). In this protocol, a secret key is established between Alice and Bob with the help of an untrusted third-party called Eve. We study how one can suitably mount a symmetric incoherent eavesdropping strategy on MDI QKD considering that Eve will not be honest. Similar to the idea of Fuchs et al. (1997), we show that inducing a disturbance $D$ on the Alice’s (or Bob’s) qubit, Eve can guess Alice’s (or Bob’s) bit with probability $\frac{1}{2} + \sqrt{D(1-D)}$. If Eve likes to guess both the bits of Alice and Bob at the same time, then the success probability becomes $\frac{1}{4} + \frac{\Delta}{2} + \frac{\sqrt{\Delta}}{2}$ with an equivalent disturbance of $\Delta = 2D(1-D)$ between Alice and Bob. We also study how well Eve can guess whether an error has been introduced between Alice and Bob due to her interaction. While in BB84, Eve can identify with certainty whether an error has been introduced because of eavesdropping, for MDI-QKD, she can only guess it with probability $\frac{1}{2} + 2D(1-D)$.

Keywords BB84 · Eavesdropping · Entanglement Swapping · Quantum Key Distribution · Measurement Device Independence · Quantum Cryptography

Mathematics Subject Classification (2010) 94A60 · 81P94

CR Subject Classification (2012) Quantum communication and cryptography
1 Introduction

The idea of BB84 [1,2] protocol was initiated by Bennet and Brassard based on the seminal observation by Wiesner [15]. The BB84 protocol is used by two parties called Alice and Bob to settle on a secret classical bit-string over an insecure quantum channel where Eve, the eavesdropper, can have access. The argument behind the security of the BB84 [2] protocol is guided by the physical law that one cannot clone an unknown quantum state perfectly. This leads to the understanding that if one wants to distinguish two non-orthogonal quantum states, then obtaining any information is only possible at the expense of introducing disturbance in the state(s). As the BB84 protocol [2] exploits non-orthogonal quantum states, any kind of eavesdropping induces disturbance to the qubits communicated from Alice to Bob. Further, the disturbance caused due to Eve’s interaction can be modeled as a Binary Symmetric Channel (BSC) between Alice and Bob with some error. We use the idea of eavesdropping following [8,9] against MDI-QKD and then exploit elementary combinatorial techniques to obtain our results.

There are several variants of the traditional BB84 protocol [3–7,10–13] that received attention in literature. Given that BB84 is the most celebrated quantum cryptographic protocol, proposals for BB84 variants are of active interest and they come from different motivations. For example, the semiquantum protocol [4,5] considers that Bob will only have restricted capability that he can measure in a specific basis. The two-way protocol [11,12] is motivated from the idea that Alice and Bob will not discuss the basis in public for the qubits that correspond to secret key bits. The very recent proposals [6,10] are motivated from resistance against side channel attacks where they allow an untrusted party in the protocol.

To resist detector side channel attacks, Measurement Device Independent (MDI) Quantum Key Distribution (QKD) idea has been presented very recently in [10]. We will show how the idea of symmetric incoherent eavesdropping of [9] can be suitably modified to be accommodated in this scenario. As this proposal is very recent, to the best of our knowledge, such attack has not yet been studied.

In MDI QKD [10], Alice and Bob need not measure any qubit, and all the measurements are executed at Eve’s end, an untrusted third-party. Thus, for eavesdropping strategies, it is natural to consider that Eve herself will try to gather information about the secret key while assisting Alice and Bob. That is why, this attack can be termed as third-party attack. While the idea of [6] uses entanglement swapping [16] for building the protocol, it is interesting to note that we exploit this for third-party eavesdropping against MDI QKD [10]. The application of entanglement swapping is evident in such protocols (either in design or in analysis) due to the involvement of the third-party.

In this paper, we build on the basic idea of eavesdropping strategy explained in [9] on the traditional BB84 [2] protocol. In MDI QKD [10], Alice and Bob send their respective qubits to Eve. Eve, while pretending to support the protocol, may not be honest and may try to extract some information regarding the bits to be decided by Alice and Bob. In quantum scenario, any attempt by Eve, to interact with the qubits sent by Alice and Bob and further to extract additional information, will induce an error between Alice and Bob. We first show that an eavesdropping strategy similar to [9] can be mounted on the qubits of either Alice or Bob so that introducing a disturbance $D$, Eve can guess the respective bit with a success probability $\frac{1}{2} + \sqrt{D(1-D)}$. Further, we show that the attack can be
extended on both the qubits of Alice and Bob to obtain a success probability
\[ \frac{1}{4} + \frac{1}{2} + \sqrt{2} \] with an equivalent disturbance \( \Delta = 2D(1 - D) \). We note that Eve’s capability to identify the bits where error had been introduced (or not) is reduced in MDI QKD as compared to BB84.

2 Description of MDI QKD [10]

To understand this algorithm, we need to use Bell states. These are two-qubit entangled states that can form orthogonal basis. The four Bell states can be written as
\[ |\Phi^\pm \rangle = \frac{1}{\sqrt{2}}(|00 \rangle \pm |11 \rangle),
|\Psi^\pm \rangle = \frac{1}{\sqrt{2}}(|01 \rangle \pm |10 \rangle) \].

<table>
<thead>
<tr>
<th>Qubits sent by Alice</th>
<th>Qubits sent by Bob</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 1 Alice and Bob create random bit strings at their ends and encodes the bits in either Z or X basis randomly and send those to Eve. 2 Eve receives each pair of qubits (one from Alice and one from Bob) and measures them in Bell basis. The detection results are publicly announced. 3 for the cases where the basis of Bob and Alice match do 4 if the qubits of Alice and Bob are in Z basis and the measurement results at Eve are |\Psi^\pm \rangle then 5 One of Alice or Bob has to flip the bit; 6 if the qubits of Alice and Bob are in X basis and the measurement result at Eve is |\Phi^- \rangle or |\Psi^- \rangle then 7 One of Alice or Bob has to flip the bit; 8 Error estimation, information reconciliation (using error correcting codes) and privacy amplification are performed by Alice and Bob on the bits at their ends to obtain the final shared key bits.

\textbf{Algorithm 1: A brief description of MDI QKD [10].}

a In the actual implementation, Eve can identify only two (\(|\Psi^\pm \rangle\)) of the four Bell states and that is claimed to be enough for the security proof to go through [10]. Our analysis will also go through in a similar manner in such a scenario.

<table>
<thead>
<tr>
<th>Qubits sent by Alice</th>
<th>Probability (Eve’s end)</th>
<th>Qubits sent by Bob</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>(\frac{3}{2} )</td>
<td>(\frac{1}{2} )</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(\frac{1}{2} )</td>
<td>(\frac{1}{2} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(\frac{1}{2} )</td>
<td>(\frac{1}{2} )</td>
<td>0</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>(\frac{3}{2} )</td>
<td>0</td>
<td>(\frac{1}{2} )</td>
</tr>
<tr>
<td>+</td>
<td>–</td>
<td>0</td>
<td>(\frac{1}{2} )</td>
<td>0</td>
</tr>
<tr>
<td>–</td>
<td>+</td>
<td>(\frac{1}{2} )</td>
<td>(\frac{1}{2} )</td>
<td>0</td>
</tr>
<tr>
<td>–</td>
<td>–</td>
<td>(\frac{3}{2} )</td>
<td>0</td>
<td>(\frac{1}{2} )</td>
</tr>
</tbody>
</table>

\textbf{Table 1} Different cases in MDI QKD [10].
The untrusted third-party Eve measures the states received from Alice and Bob in this basis and informs the measurement result back to them. Towards eavesdropping, we will also study some other measurements by Eve on the qubits through which she will interact with the qubits sent by Alice and Bob. For such purposes, based on the public discussion between Alice and Bob, Eve will measure the qubits with her in proper basis. Before proceeding further, let us first explain the strategy of [10] in Algorithm 1.

We present Table 1 for understanding all the cases. When Alice and Bob generate qubits in different bases then those pairs of qubits are discarded and thus this is not shown in the table.

3 Attack Model

The eavesdropper can work on each individual qubit as well as on a set of qubits taken together. We study the first one that is called the incoherent attack, while the second one is known as the coherent attack. Another interesting issue in specifying the eavesdropping scenario is whether there will be equal error probability at Bob’s end corresponding to different bases. If this is indeed equal, then we call it symmetric and that is what we concentrate on here. It creates certain constraint on Eve in terms of extracting information from the communicated qubits. That is, as far as Alice and Bob are concerned, the interference by Eve will produce a binary symmetric channel in each case with an error probability that we will denote by $D$. There is also another model where this is not equal and then we call the eavesdropping model as asymmetric. Different error rates for different bases would be a clear indication to Alice and Bob that an eavesdropper (Eve) is interfering in the communication line. One may refer to [8] for details on this and it has been commented in the same paper that given any asymmetric attack (coherent or incoherent), one can always get a symmetric attack that can match the results of the non-symmetric strategy.

Fuchs et al. (Phy. Rev. A, 1997) [9] presented an optimal eavesdropping strategy on the four-state BB84 protocol with the qubits in $Z = \{\ket{0}, \ket{1}\}$ and $X = \{+ = \frac{\ket{0} + \ket{1}}{\sqrt{2}}, - = \frac{\ket{0} - \ket{1}}{\sqrt{2}}\}$ basis. Later, Bruß (Phys. Rev. Lett., 1998) [7] described the use of the basis $\left\{ \frac{\ket{0} + i \ket{1}}{\sqrt{2}}, \frac{\ket{0} - i \ket{1}}{\sqrt{2}} \right\}$ $(i = \sqrt{-1})$ along with the above two to show that the BB84 protocol with three conjugate bases (six-state protocol) provides improved security. In [14], a cryptanalytic view has been taken and it is shown that if one considers Eve’s success probability in guessing the secret key instead of the mutual information between Alice and Eve, then the security levels of the protocols have different interpretations depending on the ranges of parameters used. Motivated by [14], in this paper we take the same cryptanalytic view and consider Eve’s success probability as a measure of the protocol’s strength.

Without loss of generality, we assume throughout the paper that the qubits are sent in $Z$ basis. The analysis for the $X$ basis follows from symmetry.

3.1 Eavesdropping against BB84

Let us briefly describe the attack of [9] against BB84. Consider that Eve interacts with the qubit sent by Alice to Bob. We assume that Eve has a two-qubit initial
state |W⟩_A. The unitary interactions at Eve’s end can be written as

\[
U|0⟩_A|W⟩_A = \sqrt{1 - D}|0⟩_A|E_{00}⟩_A + \sqrt{D}|1⟩_A|E_{01}⟩_A,
U|1⟩_A|W⟩_A = \sqrt{1 - D}|1⟩_A|E_{11}⟩_A + \sqrt{D}|0⟩_A|E_{10}⟩_A.
\]

(1)

where \(D\) is the disturbance and \(1 - D\) is the fidelity and \(E_{pq}\) is the state of Eve’s ancilla qubits after the interaction. This is similar to the idea of BSC in the sense that if Alice encodes 0 or 1 by a qubit and send that to Bob, then due to Eve’s interaction, an error probability \(D\) is introduced in the channel. Naturally, Eve should be able to learn some information at the expense of introducing the error in between Alice and Bob.

If we rewrite the interactions expressed in [9, Equations 50-51] in our notation, we obtain the following expressions for \(|E_{pq}\)’s.

\[
|E_{00}⟩_A = \sqrt{1 - D}|00⟩ + |11⟩, \\
|E_{01}⟩_A = \sqrt{1 - D}|01⟩ - |10⟩, \\
|E_{10}⟩_A = \sqrt{1 - D}|11⟩ + |00⟩, \\
|E_{11}⟩_A = \sqrt{1 - D}|01⟩ + |10⟩.
\]

(2)

Eve waits till Alice and Bob publicly share their basis and then measures the pair of qubits at her end in \(|00⟩, |01⟩, |10⟩, |11⟩\) basis. If Eve obtains \(|ij⟩\), then she guesses that Alice has sent the qubit \(|i⟩\) and Bob has received \(|j⟩\). It has been shown [9,8] that Eve can correctly guess the qubit sent by Alice and received by Bob with probability \(\frac{1}{2} + \sqrt{D(1 - D)}\).

For MDI QKD, we consider two attack models. In the first model, Eve eavesdrops only on one side, and in the second model, she eavesdrops on both the sides.

3.2 Eavesdropping against MDI QKD on one side

Let us first consider that Eve interacts with the qubit sent by Alice only and does not disturb the qubit communicated by Bob. The unitary interactions at Eve’s end can be written as in (1).

Now consider the case when both Alice and Bob send 0. The overall state at Eve’s end is given by

\[
\begin{align*}
&\left(\sqrt{1 - D}|0⟩_A|E_{00}⟩_A + \sqrt{D}|1⟩_A|E_{01}⟩_A\right)|0⟩_B \\
&= \sqrt{\frac{1 - D}{2}}|E_{00}⟩_A|φ^+⟩_{AB} + \sqrt{\frac{1 - D}{2}}|E_{00}⟩_A|φ^−⟩_{AB} \\
&+ \sqrt{\frac{D}{2}}|E_{01}⟩_A|ψ^+⟩_{AB} - \sqrt{\frac{D}{2}}|E_{01}⟩_A|ψ^−⟩_{AB}.
\end{align*}
\]

(3)
Similarly, when Alice sends 0 and Bob sends 1, the overall state at Eve’s end is given by

\[
(\sqrt{1-D}\ket{0}_A\ket{E_{00}}_A + \sqrt{D}\ket{1}_A\ket{E_{01}}_A)\ket{1}_B = \sqrt{\frac{1-D}{2}}\ket{E_{00}}_A\ket{\psi^+}_{AB} + \sqrt{\frac{1-D}{2}}\ket{E_{00}}_A\ket{\psi^-}_{AB} + \sqrt{\frac{D}{2}}\ket{E_{01}}_A\ket{\phi^+}_{AB} - \sqrt{\frac{D}{2}}\ket{E_{01}}_A\ket{\phi^-}_{AB}.
\]

When Alice sends 1 and Bob sends 0, the overall state at Eve’s end is given by

\[
(\sqrt{1-D}\ket{1}_A\ket{E_{11}}_A + \sqrt{D}\ket{0}_A\ket{E_{10}}_A)\ket{0}_B = \sqrt{\frac{1-D}{2}}\ket{E_{11}}_A\ket{\psi^+}_{AB} - \sqrt{\frac{1-D}{2}}\ket{E_{11}}_A\ket{\psi^-}_{AB} + \sqrt{\frac{D}{2}}\ket{E_{10}}_A\ket{\phi^+}_{AB} + \sqrt{\frac{D}{2}}\ket{E_{10}}_A\ket{\phi^-}_{AB}.
\]

When both Alice and Bob send 1, the overall state at Eve’s end is given by

\[
(\sqrt{1-D}\ket{1}_A\ket{E_{11}}_A + \sqrt{D}\ket{0}_A\ket{E_{10}}_A)\ket{1}_B = \sqrt{\frac{1-D}{2}}\ket{E_{11}}_A\ket{\phi^+}_{AB} - \sqrt{\frac{1-D}{2}}\ket{E_{11}}_A\ket{\phi^-}_{AB} + \sqrt{\frac{D}{2}}\ket{E_{10}}_A\ket{\psi^+}_{AB} + \sqrt{\frac{D}{2}}\ket{E_{10}}_A\ket{\psi^-}_{AB}.
\]

### 3.3 Eavesdropping against MDI QKD on both the sides

We assume that Eve has two separate two-qubit initial states \(\ket{W}_A\) and \(\ket{W}_B\) respectively to interact with the qubits sent by Alice and Bob respectively. The unitary interactions at Eve’s end can be written as

\[
U\ket{0}_A\ket{W}_A = \sqrt{1-D}\ket{0}_A\ket{E_{00}}_A + \sqrt{D}\ket{1}_A\ket{E_{01}}_A = \ket{\gamma}_A,
\]

\[
U\ket{1}_A\ket{W}_A = \sqrt{1-D}\ket{1}_A\ket{E_{11}}_A + \sqrt{D}\ket{0}_A\ket{E_{10}}_A = \ket{\tau}_A,
\]

\[
U\ket{0}_B\ket{W}_B = \sqrt{1-D}\ket{0}_B\ket{E_{00}}_B + \sqrt{D}\ket{1}_B\ket{E_{01}}_B = \ket{\gamma}_B,
\]

\[
U\ket{1}_B\ket{W}_B = \sqrt{1-D}\ket{1}_B\ket{E_{11}}_B + \sqrt{D}\ket{0}_B\ket{E_{10}}_B = \ket{\tau}_B.
\]

Note that for each \(p, q \in \{0, 1\}\), \(\ket{E_{pq}}_B\) is exactly the same as \(\ket{E_{pq}}_A\) as given in Equation (2). Here \(\ket{\gamma}_A, \ket{\gamma}_B, \ket{\tau}_A, \ket{\tau}_B, i \in \{0, 1\}\), are three-qubit entangled states consisting of Alice’s (Bob’s) qubit and Eve’s two qubits in each case. Here we exploit the idea of entanglement swapping [16] and we get the following states for different cases.
Consider the case when both Alice and Bob send 0. The overall state at Eve’s end is given by
\[
U|0\rangle_A |W\rangle_A \otimes U|0\rangle_B |W\rangle_B = (1 - D)|0\rangle_A |0\rangle_B |E_{00}\rangle_A |E_{00}\rangle_B + \sqrt{D(1 - D)}|1\rangle_A |0\rangle_B |E_{01}\rangle_A |E_{00}\rangle_B + \sqrt{D(1 - D)}|0\rangle_A |1\rangle_B |E_{00}\rangle_A |E_{01}\rangle_B
\]
\[
= \left( \frac{1 - D}{\sqrt{2}} |F_{0000}\rangle + \frac{D}{\sqrt{2}} |F_{0101}\rangle \right) \Phi^+_{AB} + \left( \frac{1 - D}{\sqrt{2}} |F_{0000}\rangle - \frac{D}{\sqrt{2}} |F_{0101}\rangle \right) \Phi^-_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{0100}\rangle + \sqrt{\frac{D(1 - D)}{2}} |F_{0001}\rangle \right) \Psi^+_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{0100}\rangle - \sqrt{\frac{D(1 - D)}{2}} |F_{0001}\rangle \right) \Psi^-_{AB}
\] (7)

where \(|F_{pqrs}\rangle = |E_{pq}\rangle_A |E_{rs}\rangle_B\). Similarly, when Alice sends 0 and Bob sends 1, Eve’s state is
\[
= \left( \frac{1 - D}{\sqrt{2}} |F_{0110}\rangle + \frac{D}{\sqrt{2}} |F_{1011}\rangle \right) \Phi^+_{AB} + \left( \frac{1 - D}{\sqrt{2}} |F_{0011}\rangle - \frac{D}{\sqrt{2}} |F_{0110}\rangle \right) \Psi^-_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{0111}\rangle + \sqrt{\frac{D(1 - D)}{2}} |F_{0011}\rangle \right) \Phi^+_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{0111}\rangle - \sqrt{\frac{D(1 - D)}{2}} |F_{0011}\rangle \right) \Phi^-_{AB}
\] (8)

Again, when Alice sends 1 and Bob sends 0, Eve’s state is
\[
= \left( \frac{1 - D}{\sqrt{2}} |F_{1001}\rangle + \frac{D}{\sqrt{2}} |F_{1100}\rangle \right) \Phi^+_{AB} + \left( \frac{1 - D}{\sqrt{2}} |F_{1001}\rangle - \frac{D}{\sqrt{2}} |F_{1100}\rangle \right) \Psi^-_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{1000}\rangle + \sqrt{\frac{D(1 - D)}{2}} |F_{1101}\rangle \right) \Phi^+_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{1000}\rangle - \sqrt{\frac{D(1 - D)}{2}} |F_{1101}\rangle \right) \Phi^-_{AB}
\] (9)

Finally, when both Alice and Bob send 1, Eve’s state is
\[
= \left( \frac{1 - D}{\sqrt{2}} |F_{1111}\rangle + \frac{D}{\sqrt{2}} |F_{1010}\rangle \right) \Phi^+_{AB} + \left( \frac{1 - D}{\sqrt{2}} |F_{1111}\rangle - \frac{D}{\sqrt{2}} |F_{1010}\rangle \right) \Phi^-_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{1011}\rangle + \sqrt{\frac{D(1 - D)}{2}} |F_{1110}\rangle \right) \Psi^+_{AB} + \left( \sqrt{\frac{D(1 - D)}{2}} |F_{1011}\rangle - \sqrt{\frac{D(1 - D)}{2}} |F_{1110}\rangle \right) \Psi^-_{AB}
\] (10)

We have the following result about the disturbance observed by Alice and Bob.

**Proposition 1** When Eve eavesdrops on both the sides, the effective disturbance at Alice and Bob’s end is given by \(\Delta(D) = 2D(1 - D)\).
Proof} Eve measures the qubits of Alice and Bob in the Bell basis and sends the result to both Alice and Bob. When Alice and Bob sends bits in $Z$ basis and the result is $\Phi_{AB}^\pm$, they keep their bits; whereas if the result is $\Psi_{AB}^\pm$, one of them flips the bit. From Equations (7), (8), (9) and (10), the result follows. Note that for Equations (7) and (10), an error corresponds to the measurement output being $\Psi_{AB}^\pm$ and for Equations (8) and (9), an error corresponds to the measurement output being $\Phi_{AB}^\pm$.

\[\square\]

4 Eve’s Success Probability

According to the MDI QKD protocol described in Section 2, Eve first measures the qubits of Alice and Bob in the Bell basis and sends the result to both Alice and Bob. In our attack model, she applies the identity operators on the qubits at her disposal during this measurement. Let $M$ denote the state of Eve’s ancilla qubits after the measurement. After public communication and key establishment between Alice and Bob, Eve would again measure the state at her disposal in the computational basis. We reiterate that we only consider the case when Alice and Bob’s bases match, and without loss of generality we consider when both used $Z$ basis. Let $V$ denote the state of Eve’s ancilla qubits after the second measurement. Informally, we call it the measurement outcome in the computational basis. Let $A \in \{0, 1\}$ and $B \in \{0, 1\}$ denote the bits sent by Alice and Bob respectively. Let the guesses of Eve about what Alice and Bob sent be denoted by $G_A \in \{0, 1\}$ and $G_B \in \{0, 1\}$ respectively.

Analogous to [14, Theorem 2], we can write the following result for the optimal decision of Eve.

**Proposition 2** Given an output $v$ from the measurement by Eve in the computational basis, her optimal decision is given by

\[S_{\text{opt}}(v) = \arg\max_{a,b} P(A = a, B = b \mid V = v),\]

and the corresponding optimal success probability is given by

\[P_{\text{opt}}(\text{success}) = \sum_v \max_{a,b} P(A = a, B = b, V = v),\]

where the notation $\arg\max_{a,b}$ denotes the particular tuple $(a_m, b_m)$ which maximizes the above conditional probability across all pairs $(a, b) \in \{0, 1\}^2$.

Basically, in each case, Eve’s optimal strategy would be to maximize the joint probability of what she observed and what Alice and Bob have sent and this maximum probability gives her optimal success probability.

4.1 Success probability for eavesdropping on one side

The attack model for this case has already been described in Section 3.2. Note that here $V \in \{0, 1\}^2$. We have the following result.
Table 2  Likelihood $P(V = v | A = a, B = b)$ for one-sided eavesdropping.

<table>
<thead>
<tr>
<th>$V$</th>
<th>$A = 0$, $B = 0$</th>
<th>$A = 0$, $B = 1$</th>
<th>$A = 1$, $B = 0$</th>
<th>$A = 1$, $B = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
</tr>
<tr>
<td>01</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
</tr>
<tr>
<td>10</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
<td>$Dd'$</td>
</tr>
<tr>
<td>11</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
<td>$(1 - D)d'$</td>
</tr>
</tbody>
</table>

**Lemma 1** The likelihoods $P(V = v | A = a, B = b)$, for $v \in \{0,1\}^2$, $a, b \in \{0,1\}$, are given as in Table 2, where $d = \frac{1}{2} + \sqrt{D(1 - D)}$ and $d' = \frac{1}{2} - \sqrt{D(1 - D)}$.

**Proof** The likelihood $P(V = v | A = a, B = b)$ can be computed as

$$\sum_{p,q} P(M = |E_{pq}| | A = a, B = b)P(V = v | A = a, B = b, M = |E_{pq}|).$$

In the above expression, the values of $P(M = |E_{pq}| | A = a, B = b)$ can be obtained directly from Equations (3), (4), (5) and (6) and $P(V = v | A = a, B = b, M = |E_{pq}|)$ can be directly obtained from the corresponding expressions in Equation (2). \(\square\)

Using Bayes’ theorem, one can compute the table for the values of $P(A = a, B = b | V = v)$ to obtain optimal decision.

Below we derive the success probability of Eve due to the above strategy.

**Theorem 1** The optimal success probability of Eve in guessing the bit sent by Alice is given by $\frac{1}{2} + \sqrt{D(1 - D)}$.

**Proof** The strategy of Eve will be as follows. If Eve observes $V = 00$ or $10$ (respectively $V = 01$ or $11$), then she will consider Alice sent 0 (respectively 1). In this case the probability of Eve in correctly guessing Alice’s bit is $d = \frac{1}{2} + \sqrt{D(1 - D)}$. That gives the proof. \(\square\)

The following result is immediate as we do not have knowledge about Bob’s bit other than the random guess, i.e., Eve can guess Bob’s bit with probability $\frac{1}{4}$ only.

**Corollary 1** Success probability for guessing both Alice’s and Bob’s bit is $P_1(D) = \frac{1}{2}d = \frac{1}{4} + \frac{1}{2}\sqrt{D(1 - D)}$.

4.2 Success probability for eavesdropping on two sides

Refer to Section 3.3 for the exact model of the attack. Note that here we have $V \in \{0,1\}^4$. We begin by computing the likelihoods.

**Lemma 2** The likelihoods $P(V = v | A = a, B = b)$, for $v \in \{0,1\}^4$, $a, b \in \{0,1\}$, are given as in Table 3, where $d_\pm = (\sqrt{1 - D} \pm \sqrt{D})^2$ and $d_2 = (1 - 2D)^2$.

**Proof** The likelihood $P(V = v | A = a, B = b)$ can be computed as

$$\sum_{p,q,r,s} P(M = |F_{pqrs}| | A = a, B = b)P(V = v | A = a, B = b, M = |F_{pqrs}|).$$

In the above expression, the values of $P(M = |F_{pqrs}| | A = a, B = b)$ can be obtained directly from Equations (7), (8), (9) and (10) and $P(V = v | A = a, B = b, M = |F_{pqrs}|)$ can be directly obtained from the corresponding expressions of $|F_{pqrs}| = |E_{pq}A|E_{rs}B$ via Equation (2). \(\square\)
In this case also, using Bayes’ theorem, one can calculate the values of \( P(A = a, B = b | V = v) \). Theorem 2 gives the success probability of Eve due to the above strategy.

**Theorem 2** The optimal success probability of Eve in guessing a pair of bits sent by Alice and Bob is given by

\[
P_2(D) = \frac{1}{4} + D(1 - D) + \sqrt{D(1 - D)}.
\]

**Proof** We need to multiply each entry in Table 3 with \( \frac{1}{4} \) to find out the joint probabilities. According to Proposition 2, for each possible outcome \( v \), we need to find the maximum such joint probability and sum them up. In each row, we have three probabilities to compare, namely, \( d_+ = (\sqrt{1 - D} + \sqrt{D})^4 \), \( d_- = (\sqrt{1 - D} - \sqrt{D})^4 \), and \( d_2 = (1 - 2D)^2 \). Note that \( d_\pm = (\sqrt{1 - D} \pm \sqrt{D})^4 = 1 + 4D(1 - D) + 4\sqrt{D(1 - D)}. \) For \( 0 \leq D \leq 1 \), the last term is non-negative and so \( d_+ \geq d_- \). Again, \( d_2 = 1 - 4D + 4D^2 = 1 - 4D(1 - D) \leq 1 + 4D(1 - D) \leq d_+ \). Adding up the maximum joints for each row, we get \( \frac{1}{4} d_+ \) that gives the required expression. \( \square \)

Thus, from Proposition 1 and Theorem 2, we get the following result.

**Corollary 2** Introducing a disturbance \( \Delta \), the optimal success probability of Eve in guessing a pair of bits sent by Alice and Bob is given by

\[
\frac{1}{4} + \frac{1}{2} + \sqrt{\frac{1}{2}}.
\]

In Table 4, we list Eve’s optimal guesses \( G_A \) and \( G_B \) of what Alice and Bob sent respectively corresponding to her different measurement outcomes \( V \).

### 4.3 Guessing the location of the errors

Consider the eavesdropping against BB84 as in (1). One may easily note that if Alice sends \( |0\rangle \), then Eve will obtain either \( |E_{00}\rangle \) or \( |E_{01}\rangle \). The measurement is in \( |00\rangle, |01\rangle, |10\rangle, |11\rangle \) basis. Given the forms of \( |E_{pq}\rangle \) as in (2), after measurement if Eve observes \( |00\rangle \) or \( |11\rangle \) basis, then she knows that Bob obtained \( |0\rangle \), i.e., no error.
has been introduced. On the other hand, after measurement, if Eve observes |01⟩ or |10⟩ basis, then she knows that Bob obtained |1⟩, i.e., error has been introduced. In this manner, it can be checked that in BB84 protocol, Eve can decide with certainty whether her interaction has introduced an error or not.

We first note that in case of one-sided attack on MDI-QKD, Eve cannot get this advantage. This is because, in this case Alice and Bob produce the bits independently and looking at Alice’s bit, it is not possible to know what happens in case of Bob’s bit. Next we analyze the case for two-sided eavesdropping.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$G_A$</th>
<th>$G_B$</th>
<th>$P(A, B, G_A, G_B)$</th>
<th>Error Guessed by Eve correctly</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_+$</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>$\frac{1}{\sqrt{2}}d_-$</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 5 Studying the probability of Eve’s guess in correctly identifying whether error has occurred between Alice and Bob.

After estimating the bits of both Alice and Bob, and knowing the Bell states, Eve can probabilistically guess in which bit the error occurs. Let us now describe how this works.

One should have a look at the Table 1 also for the scenario when there is no eavesdropping. In our attack model, when the Bell measurement gives $\Phi^\pm$, none of Alice and Bob flips the bits. Hence, an error is introduced if one of Alice had sent 0 and the other had sent 1. So Eve can identify that an error has occurred if she
guesses one of Alice and Bob’s bits to be 0 and the other to be 1. Again, when the Bell measurement gives \( \psi^k \), one of Alice and Bob flips the bit. Hence, an error is introduced if both Alice and Bob had sent 0 or 1. So Eve can correctly identify the error if she guesses both Alice and Bob’s bits as 0 or 1. Thus, in both the cases, Eve’s can identify the error if she guesses both the bits correctly or both the bits wrongly. In Theorem 3, we derive the Probability of Eve’s correctly identifying the error at Alice and Bob’s ends.

Theorem 3  
Eve can guess whether an error has been introduced between Alice and Bob or not with probability \( \frac{1}{2} + 2D(1 - D) \).

Proof

Eve can correctly guess the error introduced between Alice and Bob when she guesses either both of the bits correctly or both of the bits incorrectly. Hence this probability is given by

\[
\sum_{a,b} \left( P(A = a, B = b, G_A = a, G_B = b) + P(A = a, B = b, G_A = \bar{x}, G_B = \bar{b}) \right),
\]

where \( \bar{x} \) denotes the bit-compliment of \( x \). From Table 4, we see a 1-to-1 correspondence between Eve’s guess \((G_A, G_B)\) and the measurement outcome \( V \). Let \( v(x, y) \) denote the outcome corresponding to the guess \( G_A = x, G_B = y \). Thus, the above probability reduces to

\[
\sum_{a,b} \left( P(A = a, B = b, V = v(a, b)) + P(A = a, B = b, V = v(\bar{x}, \bar{b})) \right).
\]

From Table 3 we can easily compute the required joint probabilities and these are shown in Table 5. Thus the probability is \( \frac{1}{16}(d_+ + d_-) = \frac{1}{2} + 2D(1 - D) \). \( \square \)

5 Discussion and Conclusion

In this paper, we have studied eavesdropping strategy on a recently proposed variant of BB84, which is refereed to as MDI QKD [10]. This variant is motivated from resistance against side channel attacks that relies on measurement device independence for the communicating parties. We analyze an existing eavesdropping strategy [9] against BB84 [2] on this MDI QKD protocol and critically compare the effectiveness of the strategy against BB84 and MDI QKD.

The difference between the attacker’s success probability and the probability of random guess gives the attacker’s advantage. In our analysis, the probability of random guess is \( \frac{1}{4} \). Thus, Eve’s advantages for the one-sided and the two-sided attacks are respectively given by

\[
A_1(D) = P_1(D) - \frac{1}{4} = \frac{1}{2}\sqrt{D(1 - D)}, \quad \text{and} \quad A_2(D) = P_2(D) - \frac{1}{4} = D(1 - D) + \sqrt{D(1 - D)}.
\]

In the analysis of both one-sided and two-sided eavesdropping, Eve measures all the ancilla qubits at her disposal. However, Eve may not concentrate on all the qubits. For the one-sided case, she needs to measure only the second qubit and for the two-sided case she needs to measure only the second and the fourth qubits. By forming likelihood tables corresponding to these partial measurements, it can be immediately shown that Eve arrives at the same success probabilities \( P_1(D) \) and \( P_2(D) \) respectively for the two attacks. Measuring the second one among the two qubits of the ancilla has already known in literature [9] and that can be followed here as well.

In two-sided eavesdropping, if Eve is interested to know only Alice’s bit, she would form a likelihood table similar to Table 3, but for the probabilities \( P(V = \)
|A = a). By summing up the maximum joint probabilities for each row of this new table, one can easily see that the optimal success probability to guess only Alice’s bit for two-sided eavesdropping case is given by is \( \frac{1}{2} + \sqrt{D(1-D)} \), which is the same for one-sided eavesdropping as shown in Theorem 1. The same success probability is obtained for guessing only Bob’s bit in two-sided eavesdropping. It is tempting to multiply the success probabilities of two independent one-sided attacks to achieve \((\frac{1}{2} + \sqrt{D(1-D)})^2 = \frac{1}{4} + D(1-D) + \sqrt{D(1-D)}\) which is the success probability for two-sided attack. However, we would like to emphasize that in presence of entanglement swapping, such independence should not be presumed. Rather, our direct derivation without any assumption brings forth an interesting observation: performing the attack on only one side is equivalent to performing the attack on both sides and then observing the result for only one side.

Apparently, the attack of [9] against BB84 protocol is sharper in the sense that both the bits of Alice and Bob could have been guessed with a probability \( d = \frac{1}{2} + \sqrt{D(1-D)} \) which is greater than \( P_1(D) \) or \( P_2(D) \). However, such a comparison is not fair, since in the attack of [9], Alice is the sender, Bob is the receiver and Eve’s only goal is to decide the bit that Alice has sent. Thus she has to guess only one bit at a time. Whereas in the current scenario, she has to guess two bits, one sent by Alice and the other sent by Bob. Definitely the latter is a more challenging case having twice as large a sample space as the former. However, if Eve does not want to know what Alice and Bob had sent individually, but she is interested only in the secret key bit established between Alice and Bob, then for MDI QKD she will perform just one-sided attack and would guess the secret key based on whether one of Alice and Bob flips the bit or not.

However, there is one aspect in which the eavesdropper has an extra advantage in traditional BB84 over MDI QKD. In BB84, Whenever Eve’s post-measurement state is \( |01\rangle \) or \( |10\rangle \), she knows with probability 1 that an error has occurred (i.e., Alice and Bob’s bits do not match), and when the state is \( |00\rangle \) or \( |11\rangle \), she knows with probability 1 that no error has occurred. But in MDI QKD, Eve cannot get any information about the location of the errors for one-sided attack. Even for two-sided attack, none of her post-measurement states tells her with certainty in which bits the errors are actually introduced. As Table 5 shows, she can get the additional information regarding the location of the bits where errors are actually introduced only with certain probabilities (and with an overall probability of \( \frac{1}{2} + 2D(1-D) \)). In this aspect, MDI-QKD [10] leaks less information than BB84 [2].

References

On Construction D and Related Constructions of Lattices from Linear Codes

Wittawat Kositwattanarerk · Frédérique Oggier

Received: date / Accepted: date

Abstract We investigate three closely related constructions of lattices from linear codes: the classical Construction D, Construction $\mathcal{D}$, and the recently developed Construction $\mathcal{A}'$. These constructions have been proven useful and result in efficient encoding and decoding algorithms for Barnes-Wall lattices. Here, we analyze their applications in a general setting. We show that Construction $\mathcal{D}$ produces a lattice packing if and only if the nested codes being used are closed under Schur product, thus proving the similarity of Construction D and Construction $\mathcal{D}$ when applied to Reed-Muller codes. In addition, we provide a correspondence between nested binary linear codes and codes over polynomial rings. This proves that Construction $\mathcal{A}'$ does not always produce a lattice, but any lattices constructible using Construction $\mathcal{D}$ are also constructible using Construction $\mathcal{A}'$. This result also gives a partial condition for Construction $\mathcal{A}'$ to produce a lattice.

Keywords Lattices · lattices from codes · Barnes-Wall lattices · Schur product of codes

1 Introduction

Connections between lattices and linear codes are classically studied (see e.g. [4]). Lattices constructed from codes often inherit certain properties from the underlying codes and have manageable encoding and decoding complexity [1–8]. In particular, Construction D [2] produces lattice packings from a family of nested binary linear codes and is well-known for the construction of Barnes-Wall lattices from Reed-Muller codes. This relationship is further developed by Forney [5] where Barnes-Wall lattices are constructed as a direct sum of Reed-Muller codes. This
construction has recently been adopted by Harshan, Viterbo, and Belfiore [6,7]. They also introduce Construction $A'$ as an extension of Construction $A$ to codes over polynomial rings and prove the equivalence of an encoding of Barnes-Wall lattices using Construction $A'$ and nested Reed-Muller codes, allowing them to obtain a new efficient encoder for Barnes-Wall lattices.

So far, the discussions on the above constructions were limited to Reed-Muller codes and Barnes-Wall lattices. This motivates us to address the construction of [5] and [6,7] to arbitrary codes and lattices. Here, we make clear the distinction between the classical Construction D as given in [2,4] and the construction given in [6,7] (also referred to as Construction D in the original manuscript) by referring to the latter as Construction $\overline{D}$. We demonstrate that the two constructions produce distinct lattices and provide a necessary and sufficient condition for the resulting lattices to coincide. In addition, we relate Construction $\overline{D}$ to Construction $A'$ by providing a correspondence between nested binary codes and codes over rings.

2 Definition of Construction $\overline{D}$

Let $\psi$ be the natural embedding of $\mathbb{F}_2^n$ into $\mathbb{Z}^n$, where $\mathbb{F}_2$ is the binary field. We first recall the definition of Construction D [4].

**Definition 1 (Construction D (scaled))** Let $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n$ be a family of nested binary linear codes where the minimum distance of $C_i$ is at least $4^{a-i}/\gamma$ where $\gamma = 1$ or 2. Let $k_i = \dim(C_i)$ and let $b_1, b_2, \ldots, b_{k_i}$ be a basis of $\mathbb{F}_2^n$ such that $b_1, \ldots, b_{k_i}$ span $C_i$. The lattice $\Lambda_D$ consists of all vectors of the form

$$a_{i-1} \sum_{i=0}^{a-1} 2^i \sum_{j=1}^{k_i} \alpha_{j}^{(i)} \psi(b_j) + 2^a l$$

where $\alpha_{j}^{(i)} \in \{0, 1\}$ and $l \in \mathbb{Z}^n$.

We will relax the minimum distance condition since it does not affect our discussion on constructions of lattices. The following construction is used by Forney [5] (though without special notice) and Harshan, Viterbo, and Belfiore [6,7] to construct Barnes-Wall lattices from Reed-Muller codes. We give a precise definition as follows.

**Definition 2 (Construction $\overline{D}$)** Let $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n$ be a family of nested binary linear codes. Let $\Gamma_{\overline{D}} = \psi(C_0) \oplus 2\psi(C_1) \oplus \ldots \oplus 2^{a-1}\psi(C_{a-1}) \oplus 2^a \mathbb{Z}^n$.

We will see in Example 1 that the set $\Gamma_{\overline{D}}$ itself may not necessarily be a lattice. Hence, we denote by $\Lambda_{\overline{D}}$ the smallest lattice that contains $\Gamma_{\overline{D}}$. Example 1 will also demonstrate that $\Lambda_D$ and $\Lambda_{\overline{D}}$ are two distinct lattices.

**Example 1** Consider nested binary linear codes

$C_0 \subseteq C_1 \subseteq C_2$
where
\[
C_0 = \langle (1, 1, 0, 0), (1, 0, 1, 0) \rangle, \\
C_1 = \langle (1, 1, 0, 0), (1, 0, 1, 0), (1, 0, 0, 1) \rangle, \text{ and} \\
C_2 = \mathbb{Z}_2^4.
\]

Alternatively, let \(\{b_1, b_2, b_3, b_4\}\) be a basis for \(\mathbb{Z}_2^4\) where
\[
\begin{align*}
b_1 &= (1, 1, 0, 0) \\
b_2 &= (1, 0, 1, 0) \\
b_3 &= (1, 0, 0, 1) \\
b_4 &= (1, 0, 0, 0)
\end{align*}
\]

Then,
\[
A_D = \left\{ (\alpha_1^{(0)}(1, 1, 0, 0) + \alpha_2^{(0)}(1, 0, 1, 0)) + 2(\alpha_1^{(1)}(1, 1, 0, 0) + \alpha_2^{(1)}(1, 0, 1, 0) + \alpha_3^{(1)}(1, 0, 0, 1)) + 4l \mid \alpha_i^{(j)} \in \{0, 1\}, l \in \mathbb{Z}_4 \right\}
\]

Also,
\[
\Gamma_{\mathcal{T}} = \psi(C_0) \oplus 2\psi(C_1) \oplus 4\mathbb{Z}_4^4 = \{ c_0 + 2c_1 + 4l \mid c_0 \in \psi(C_0), c_1 \in \psi(C_1), \text{ and } l \in \mathbb{Z}_4 \}.
\]

Since every entry of the vectors in \(\psi(C_0) \oplus 2\psi(C_1)\) is at most 3, every element of \(\Gamma_{\mathcal{T}}\) reduces mod 4 to an element in \(\psi(C_0) \oplus 2\psi(C_1)\). Now, since \((1, 1, 0, 0), (1, 0, 1, 0), (0, 1, 1, 0) \in \Gamma_{\mathcal{T}}\) but \((2, 0, 0, 0) = (1, 1, 0, 0) + (1, 0, 1, 0) - (0, 1, 1, 0)\) does not reduce mod 4 to an element in \(\psi(C_0) \oplus 2\psi(C_1)\), we can conclude that \((2, 0, 0, 0) \notin \Gamma_{\mathcal{T}}\) and \(\Gamma_{\mathcal{T}}\) is not a lattice. Therefore, \(\Gamma_{\mathcal{T}} \subseteq A_D\).

It is clear that \(A_D \subseteq A_{\mathcal{T}}\). Now, every element \((a_1, a_2, a_3, a_4)\) of \(A_D\) must satisfy
\[
a_1 \equiv a_2 + a_3 + a_4 \pmod{4}. \quad (1)
\]

However, since \((0, 1, 1, 0) \notin A_{\mathcal{T}}\) does not satisfy equation (1), we conclude that \(A_D \subseteq A_{\mathcal{T}}\).

Remark 1 1. Given a chain of binary linear codes \(C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^2\), the lattice \(A_D\) from Construction D depends on the choice of generators \(b_1, b_2, \ldots, b_a\), whereas the lattice \(A_{\mathcal{T}}\) from Construction D' is independent of this choice.

2. The above example can also be used to demonstrate that the classical Construction D' (see [4] for the definition) produces a lattice that is distinct from \(A_D\) and \(A_{\mathcal{T}}\). We skip the discussion on Construction D' due to lack of space.
3 Connections between Construction D and D

In this section, we give an explicit description of the lattices constructed using Construction D and further relate Construction D to Construction D using the Schur product of codes. Here, \( \oplus \) denotes addition over \( \mathbb{F}_2 \), and \( \ast \) denotes component-wise multiplication (known also as the Schur product or Hadamard product).

That is, for \( x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \mathbb{F}_2^n \), we have
\[
x \ast y := (x_1y_1, \ldots, x_ny_n) \in \mathbb{F}_2^n.
\]

It is easy to see that \( \text{supp}(x \ast y) = \text{supp}(x) \cap \text{supp}(y) \) and
\[
\psi(x) + \psi(y) = \psi(x \oplus y) + 2\psi(x \ast y).
\]

We say that a family of nested binary linear codes \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) is closed under Schur product if and only if the Schur product of any two codewords of \( C_i \) is contained in \( C_{i+1} \) for all \( i \). In other words, if \( c_i, c_j \in C_i \), then \( c_i \ast c_j \in C_{i+1} \) for all \( i = 0, \ldots, a - 1 \).

**Proposition 1** Let \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) be a family of nested binary linear codes. The lattice \( \Lambda^{\ast} \) consists of all vectors of the form
\[
\sum_{i=0}^{a-1} 2^i \sum_{c_j \in C_i} a_j^{(i)} \psi(c_j) + 2^a l
\]
where \( a_j^{(i)} \in \{0,1\} \) and \( l \in \mathbb{Z}^n \).

**Proof** Let
\[
A = \left\{ \sum_{i=0}^{a-1} 2^i \sum_{c_j \in C_i} a_j^{(i)} \psi(c_j) + 2^a l \bigg\vert a_j^{(i)} \in \{0,1\} \text{ and } l \in \mathbb{Z}^n \right\}.
\]

It is not hard to see that \( A \) is a lattice and \( A \) contains \( \Gamma^{\ast} \). Let \( A' \) be a lattice such that \( \Gamma^{\ast} \subseteq A' \), and let \( v \in A \). One may express \( v \) as
\[
\sum_{i=0}^{a-1} 2^i \sum_{c_j \in C_i} a_j^{(i)} \psi(c_j) + 2^a l
\]
where \( a_j^{(i)} \in \{0,1\} \) and \( l \in \mathbb{Z}^n \). Now, since \( 2^a l \in \Gamma^{\ast} \) and \( 2^i a_j^{(i)} \psi(c_j) \in \Gamma^{\ast} \) for all \( i \in \{0,1,\ldots,a-1\} \) and \( c_j \in C_i \), we must have \( v \in A' \). We conclude that \( A \) is the smallest lattice that contains \( \Gamma^{\ast} \) and hence is the lattice constructed from the chain \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) using Construction D.

**Remark 2** An expression (2) of an element in \( \Lambda^{\ast} \) is often not unique.

Note that Construction D depends on the chosen basis \( b_1, b_2, \ldots, b_n \) of \( \mathbb{F}_2^n \) where \( b_1, \ldots, b_k \) span \( C_i \). The following corollary states that the sum of all such lattices yields the lattice from Construction D using the same nested codes.
Corollary 1 Let ℳ be the set of all lattices constructible from a family of nested binary linear codes \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) using Construction D. Then,

\[
\bigoplus_{A_D \in \mathcal{D}} A_D = A_{\mathcal{D}}.
\]

In the following theorem, we give a sufficient and necessary condition for

\[
I_{\mathcal{D}} = \psi(C_0) \oplus 2\psi(C_1) \oplus \ldots \oplus 2^{a-1}\psi(C_{a-1}) \oplus 2^a\mathbb{Z}^n
\]

from Construction \( \mathcal{D} \) to be a lattice. In addition, when the condition is met, the resulting lattice is the same as the lattice from Construction D.

Theorem 1 Given a family of nested binary linear codes \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \), the following statements are equivalent.

1. \( I_{\mathcal{D}} \) is a lattice.
2. \( I_{\mathcal{D}} = A_{\mathcal{D}} \).
3. \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) is closed under Schur product.
4. \( I_{\mathcal{D}} = A_{\mathcal{D}} \).

Proof It is easy to see that \( 4 \Rightarrow 1 \Rightarrow 2 \). We have left to show that \( 2 \Rightarrow 3 \) and \( 3 \Rightarrow 4 \).

Suppose that \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) is not closed under Schur product. That is, there exist \( c_1, c_2 \in C_i \) such that \( c_1 \ast c_2 \notin C_{i+1} \) for some \( i \). Therefore, 

\[
2^i\psi(c_1 \ast c_2) \notin I_{\mathcal{D}}.
\]

On the other hand,

\[
2^i\psi(c_1 \ast c_2) = 2^i\psi(c_1) + 2^i\psi(c_2) - 2^i\psi(c_1 \oplus c_2) \in A_{\mathcal{D}}.
\]

Thus, we have \( I_{\mathcal{D}} \neq A_{\mathcal{D}} \).

Now, we will prove \( 3 \Rightarrow 4 \) by induction. The case \( a = 1 \) is trivial since both \( I_{\mathcal{D}} \) and \( A_{\mathcal{D}} \) coincide with the lattice constructed from \( C_0 \) using Construction A. Let \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) be a family of nested binary linear codes that is closed under Schur product, and let \( b_1, b_2, \ldots, b_n \) be a basis of \( \mathbb{F}_2^n \) such that \( b_1, \ldots, b_n \) span \( C_i \). Applying induction hypothesis to \( C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) yields

\[
I'_{\mathcal{D}} = A'_{\mathcal{D}} \quad \text{(3)}
\]

where

\[
I'_{\mathcal{D}} = 2\psi(C_1) \oplus \ldots \oplus 2^{a-1}\psi(C_{a-1}) \oplus 2^a\mathbb{Z}^n
\]

and

\[
A'_{\mathcal{D}} = \left\{ \sum_{i=1}^{a-1} 2^i \sum_{j=1}^{k_i} \alpha_j(i) \psi(b_j) + 2^a l \mid \alpha_j(i) \in \{0, 1\} \text{ and } l \in \mathbb{Z}^n \right\}.
\]

To avoid confusion, we denote by \( L \) the lattice given in (3). We now wish to show that

\[
I_{\mathcal{D}} = \psi(C_0) \oplus L = \{ \psi(c) + a \mid c \in C_0 \text{ and } a \in L \}
\]
is equal to
\[ AD = \left\{ \sum_{j=1}^{k_0} \alpha_j^{(0)} \psi(b_j) + a \left| \alpha_j^{(0)} \in \{0, 1\} \text{ and } a \in A \right. \right\}. \]

To do so, we will prove by induction that if \( c \in C_0 \) is a binary sum of \( b_{j_1}, \ldots, b_{j_s}, 1 \leq j_1, \ldots, j_s \leq k_0 \), then
\[ \psi(b_{j_1}) + \ldots + \psi(b_{j_s}) = \psi(c) + a \]
for some \( a \in A \). The case \( s = 1 \) is trivial since one may pick \( a = 0 \in A \). Let \( c \in C_0 \) be a binary sum of \( b_{j_1}, \ldots, b_{j_s}, 1 \leq j_1, \ldots, j_s \leq k_0 \). By induction hypothesis, there exists \( a' \in A \) such that
\[ \psi(b_{j_1}) + \ldots + \psi(b_{j_{s-1}}) = \psi(c') + a' \]
where \( c' \in C_0 \) is a binary sum of \( b_{j_1}, \ldots, b_{j_{s-1}} \). Now,
\[ \psi(b_{j_1}) + \ldots + \psi(b_{j_{s-1}}) + \psi(b_{j_s}) = \psi(c') + \psi(b_{j_s}) + a' \]
\[ = \psi(c' \boxplus b_{j_s}) + 2\psi(c' \ast b_{j_s}) + a' \]
\[ = \psi(c) + 2\psi(c' \ast b_{j_s}) + a'. \]

Since \( C_0 \subseteq C_1 \) is closed under Schur product, \( c' \ast b_{j_s} \in C_1 \), and so \( 2\psi(c' \ast b_{j_s}) \in A \). Letting \( a = 2\psi(c' \ast b_{j_s}) + a' \in A \), we obtain
\[ \psi(b_{j_1}) + \ldots + \psi(b_{j_s}) = \psi(c) + a \]
as desired. We can now conclude that \( \Gamma_D = A_D \), and this finishes the proof of the theorem. \( \square \)

The above theorem explains why Construction D and Construction \( \overline{D} \) have both been successful in constructing and encoding Barnes-Wall lattices. This is due to the fact that a family of Reed-Muller codes is closed under Schur product. For the same reason, Construction D produces a unique lattice despite many choices for the basis of Reed-Muller codes. This is summarized in the corollaries below.

**Corollary 2** If a family of nested binary linear codes \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) is closed under Schur product, then Construction D and \( \overline{D} \) yield the same lattice.

**Corollary 3** The Barnes-Wall lattices can be constructed using Construction \( \overline{D} \).

**Corollary 4** If a family of nested binary linear codes \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n \) is closed under Schur product, then the resulting lattices from Construction D are the same irrespective of the chosen basis \( b_1, b_2, \ldots, b_n \) of \( \mathbb{F}_2^n \) such that \( b_1, \ldots, b_{k_i} \) span \( C_i \).
4 Construction $A'$

In this section, we will consider both real and complex lattices, where a complex lattice is a $\mathbb{Z}[i]$-module over $\mathbb{C}$. Denote $\mathcal{R}$ either $\mathbb{Z}$ or $\mathbb{Z}[i]$, and let $v = 2$ if $\mathcal{R} = \mathbb{Z}$ and $v = -1 + i$ if $\mathcal{R} = \mathbb{Z}[i]$. In other words, the constant $v$ takes on different values depending on the choice of $\mathcal{R}$. Again, let $\psi$ be the natural embedding of $\mathbb{F}_2^n$ into $\mathbb{R}^n$. We rephrase Construction $D$ as follows.

**Definition 3 (Construction $D$)** A lattice $\Lambda_D$ over $\mathcal{R}$ is obtained from Construction $D$ using a family of nested binary linear codes $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_a = \mathbb{F}_2^n$ if $\Lambda_D$ is the smallest lattice that contains

$$\Gamma_D = \psi(C_0) \oplus v\psi(C_1) \oplus \ldots \oplus v^{a-1}\psi(C_{a-1}) \oplus v^a \mathbb{R}^n.$$  

Define the polynomial quotient ring $\mathcal{U}_a := \mathbb{F}_2[u]/u^a$ where $u$ is a variable. A linear code over $\mathcal{U}_a$ is a submodule of $\mathcal{U}_a^n$. The code $C$ corresponding to a generator matrix $G \in \mathcal{U}_{k \times n}$ is given by

$$C = \{uG \mid u \in \mathcal{U}_a^k\}$$

where the matrix multiplication is over the ring $\mathcal{U}_a$. One may embed $\mathcal{U}_a$ into $\mathcal{R}$ via the mapping $\Phi: \mathcal{U}_a \rightarrow \mathcal{R}$ given by

$$\Phi \left( \sum_{j=0}^{a-1} b_j u^j \right) = \sum_{j=0}^{a-1} \psi(b_j) v^j.$$  

We will also use $\Phi$ as a bit-wise embedding from $\mathcal{U}_a^n$ into $\mathcal{R}^n$. The following construction is due to [6,7].

**Definition 4 (Construction $A'$)** A lattice $\Lambda_{A'}$ over $\mathcal{R}$ is obtained from Construction $A'$ using a linear code $C$ over $\mathcal{U}_a$ if $\Lambda_{A'}$ is the smallest lattice that contains

$$\Gamma_{A'} = \Phi(C) \oplus v^a \mathbb{R}^n.$$  

The next proposition shows that there exists a correspondence from a chain of binary linear codes $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n$ to a linear code over $\mathcal{U}_a$ such that $\Gamma_D$ and $\Gamma_{A'}$ from the two constructions coincide. This would prove that $\Gamma_{A'}$ is generally not a lattice, and any lattice constructible using Construction $D$ is also constructible using Construction $A'$.

**Proposition 2** Let $C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \subseteq C_a = \mathbb{F}_2^n$ be a family of nested binary linear codes and

$$\Gamma_D = \psi(C_0) \oplus v\psi(C_1) \oplus \ldots \oplus v^{a-1}\psi(C_{a-1}) \oplus v^a \mathbb{R}^n.$$  

There exists a linear code $C$ over $\mathcal{U}_a$ such that $\Gamma_D = \Gamma_{A'}$ where

$$\Gamma_{A'} = \Phi(C) \oplus v^a \mathbb{R}^n.$$
Proof Let $k_i = \dim(C_i)$ and $k = k_{a-1} = \dim(C_{a-1})$. Let $G \in \mathbb{F}_2^{k \times n}$ be a generator matrix for $C_{a-1}$ such that the first $k_i$ rows of $G$ generates $C_i$. That is, one may write $G$ as

$$G = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{a-1} \end{bmatrix} \in \mathbb{F}_2^{k \times n}$$

where the $k_i \times n$ matrix

$$\begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_i \end{bmatrix}$$

is a generator matrix for $C_i$.

Let $C$ be a linear code over $\mathcal{U}_m$ generated by a generator matrix

$$\tilde{G} = \begin{bmatrix} G_0 \\ uG_1 \\ \vdots \\ u^{a-1}G_{a-1} \end{bmatrix} \in \mathcal{U}_a^{k \times n},$$

and let

$$\Gamma_{A'} = \Phi(C) \oplus v^a \mathcal{R}^n.$$ We will prove that $\Gamma_{A'} = \Gamma_{A'}$.

Fix an element $x \in \Gamma_{A'}$. By construction, one may express $x$ as

$$x = \psi(c_0) + v\psi(c_1) + \ldots + v^{a-1}\psi(c_{a-1}) + v^a l$$

where $c_i \in C_i$ and $l \in \mathcal{R}^n$. Let $d_i = (d_{i,1}, \ldots, d_{i,k_i}) \in \mathbb{Z}_2^{1 \times k_i}$ be a vector such that

$$d_i \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_i \end{bmatrix} = (d_{i,1}, \ldots, d_{i,k_i}) \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_i \end{bmatrix} = c_i.$$

Now, one may multiply the entries of $d_i$ by powers of $u$ and append 0 as necessary to obtain $d_i = (d_{i,1} u^i, \ldots, d_{i,k_i}, 0, \ldots, 0) \in \mathcal{U}_a^{k \times k}$ such that

$$\tilde{d}_i \tilde{G} = (d_{i,1} u^i, \ldots, d_{i,k_i}, 0, \ldots, 0) \begin{bmatrix} G_0 \\ uG_1 \\ \vdots \\ u^iG_i \\ \vdots \\ u^{a-1}G_{a-1} \end{bmatrix} = c_i u^i.$$ Thus, we have

$$(d_0 + \tilde{d}_1 + \ldots + \tilde{d}_{a-1}) \tilde{G} = c_0 + c_1 u + \ldots + c_{a-1} u^{a-1}. $$
It follows that
\[ \Phi \left( (\vec{d_0} + \vec{d_1} + \ldots + \vec{d_{a-1}})\vec{G} \right) = \psi(c_0) + \psi(c_1)v + \ldots + \psi(c_{a-1})v^{a-1}, \]
and so \( x \in \Gamma_{A'}. \) We may now conclude that \( \Gamma_D \subseteq \Gamma_{A'}. \)

On the other hand, for any \( \vec{d} \in U_a^{1 \times k}, \) the coefficient of \( u^i \) in
\[ \vec{d} \vec{G} = \vec{d} \begin{bmatrix} G_0 \\ uG_1 \\ \vdots \\ u^iG_i \\ \vdots \\ u^{a-1}G_{a-1} \end{bmatrix} \]
must be a linear combination of the rows of \( \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_i \end{bmatrix}. \) Therefore,
\[ \vec{d} \vec{G} = c_0 + c_1u + \ldots + c_iu^i + \ldots + c_{a-1}u^{a-1} \]
for some \( c_0 \in C_0, \ldots, c_{a-1} \in C_{a-1}. \) It follows that \( \Gamma_{A'} \subseteq \Gamma_D, \) and this finishes the proof of the proposition.

**Corollary 5** If a code \( C \) over \( U_a \) can be expressed as \( C_0 \oplus uC_1 \oplus \ldots \oplus u^{a-1}C_{a-1} \) where \( C_0 \subseteq C_1 \subseteq \ldots \subseteq C_{a-1} \) is closed under Schur product, then \( \Gamma_{A'} = \Phi(C) \oplus v^aR_n \) from Construction \( A' \) is a lattice.

The key ingredient of the last proposition is the fact that
\[ C_0 \oplus uC_1 \oplus \ldots \oplus u^{a-1}C_{a-1} \]
is a code over \( U_a. \) Since the converse of this statement is not true (i.e. a code over \( U_a \) is not necessarily a direct sum of binary codes), one may apply Construction \( A' \) to such code and obtain a lattice that is not constructible using Construction \( D. \) The next example will demonstrate this presumption.

**Example 2** Consider a code \( C \) of length 2 over \( U_3 \) generated by a generator matrix
\[ [1 + u 1 + u + u^2]. \]
Then, we have
\[ C = \{(0, 0), (u, u), (u^2, u^2), (u + u^2, u + u^2), (1, 1 + u^2), (1 + u, 1 + u + u^2), (1 + u + u^2, 1 + u)\}. \]
We apply Construction \( A' \) over the reals and obtain the lattice
\[ \{(0, 0), (2, 2), (4, 4), (6, 6), (1, 5), (5, 1), (3, 7), (7, 3)\} \oplus 8Z^2, \]
which is not constructible by Construction \( D \) nor Construction \( \overline{D}. \)
5 Conclusion

In this paper, we provide connections between two constructions of lattices from nested binary linear codes and a construction of lattices from codes over polynomial rings. While Construction \( D \) and Construction \( A' \) are useful in the construction of Barnes-Wall lattices, we demonstrate that they may generate nonlattice packings. We give an exact condition for Construction \( D \) to yield a lattice and further relate Construction \( D \) to Construction \( A' \). Future work involves finding a suitable condition for codes over rings to produce a lattice under Construction \( A' \). It will also be interesting to study possible properties and parameters of lattices from Construction \( D \) and Construction \( A' \).

References

Lattices from Totally Real Number Fields with Large Regulator

Soon Sheng Ong · Frédérique Oggier

Received: date / Accepted: date

Abstract We look for examples of totally real number fields with large regulator and inert small primes inside some families of cyclotomic fields. This problem is motivated by the design of wiretap codes for fast fading Rayleigh channels.

Keywords Lattices · Number Fields · Units

1 Introduction

Wiretap channels are broadcast channels that model the communication between two legitimate users, Alice and Bob, in the presence of an eavesdropper Eve. Wiretap codes aim at achieving both reliability and confidentiality via coding, by exploiting the noise that the eavesdropper experiences. Despite many works of information theoretic nature providing achievable rates and secrecy capacity results, explicit wiretap code constructions remain elusive for many classes of channels, such as additive white Gaussian noise (AWGN), fading and MIMO channels. Recently the secrecy gain and the flatness factor (see respectively [11] and [8]) have been proposed as code design criteria for AWGN wiretap channels, in order to maximize the confusion at the eavesdropper, and lattice codes satisfying the former criterion have been studied [7].

In this paper, we focus on fast fading channels instead. A code design criterion to increase the eavesdropper confusion has been computed in [2], which involves the minimization of a sum of inverse norms in number fields. This minimization has been addressed in [6] via the use of Dedekind zeta functions, and similar sums have been looked at in the context of the Diversity Multiplexing Trade-Offs (DMT)
Here instead, we translate the problem of code design for fast fading channels into finding totally real number fields with large regulator and inert small primes.

We start by recalling the code design criterion for fast fading wiretap channels in Section 2, where we motivate the need for lattices coming from totally real number fields. We then look for totally real number fields with inert small primes within some families of cyclotomic fields in Section 3. In Section 4, we further find among these totally real number fields those with bigger regulator. Since this paper only focuses on the criterion to provide confusion at the eavesdropper, the last section lists other aspects that should be incorporated to finalize the design of wiretap codes for fast fading channels.

2 Coding for the Wiretap Rayleigh Fading Channel

We consider communication over a wiretap channel, where a legitimate user Alice sends information to Bob, another legitimate user, in the presence of an eavesdropper Eve. We assume that transmission occurs over a fast Rayleigh fading channel, which models a channel where Alice, Bob and Eve all use a single antenna. We assume that perfect channel state information (CSI) is available at both receivers. Formally, Bob and Eve respectively receive the vectors $y$ and $z$ given by

$$\begin{align*}
y &= \text{diag}(h_b) x + v_b, \\
z &= \text{diag}(h_e) x + v_e,
\end{align*}$$

where $x \in \mathbb{R}^n$ is the transmitted signal, $v_b$ and $v_e$ denote the Gaussian noise at Bob, respectively Eve’s side, both with zero mean, and respective variance $\sigma_b^2$ and $\sigma_e^2$, and

$$\text{diag}(h_b) = \begin{pmatrix} |h_{b,1}| & \cdots & |h_{b,n}| \\
                        & \ddots &                        \\
|h_{b,n}| &                        & |h_{b,n}| \end{pmatrix}, \quad \text{diag}(h_e) = \begin{pmatrix} |h_{e,1}| & \cdots & |h_{e,n}| \\
                        & \ddots &                        \\
|h_{e,n}| &                        & |h_{e,n}| \end{pmatrix}$$

are the channel matrices containing the fading coefficients where $h_{b,i}, h_{e,i}$ are complex Gaussian random variables with variance $\sigma_{h,b}^2$, resp. $\sigma_{h,e}^2$, so that $|h_{b,i}|, |h_{e,i}|$ are Rayleigh distributed, $i = 1, \ldots, n$, with parameter $\sigma_{h,b}^2, \sigma_{h,e}^2$.

The goal of coding for a wiretap channel is two-fold: to ensure reliability for Bob, and confidentiality for Alice (or equivalently, confusion for Eve). The former is characterized by Bob’s probability of error, while the latter is typically interpreted via the mutual information between Alice’s message and what Eve receives.

2.1 A Code Design Criterion

In order to design wiretap codes for the fast Rayleigh fading channel, a design criterion has been proposed in [2], which we briefly recall now. We assume that Alice performs lattice encoding, that is the message intended for Bob is a vector $x \in \mathbb{R}^n$ which belongs to a lattice $\Lambda_b \subset \mathbb{R}^n$. To confuse Eve, Alice uses a standard technique for wiretap coding, that is coset coding: she chooses a sublattice $\Lambda_e$ of $\Lambda_b$, and partitions $\Lambda_b$ into a union of disjoint cosets of $\Lambda_e$. Every message sent by Alice is encoded into a coset of $\Lambda_e$, and for the actual transmission to happen, a
point in this coset is chosen uniformly at random. The intuition behind this scheme
is that coset encoding provides a labeling of the lattice points with a mixture of
random and data bits. What it should achieve is to permit Eve to decode the
random bits, but not the data bits, which is possible because of the difference of
noises between the legitimate and the eavesdropper channel. The benefit of using
lattice coset coding has been shown in [3] for the Gaussian channel. Furthermore,
to get a wiretap code design criterion, an error probability analysis is provided,
instead of the usual mutual information (though the connection between both is
made explicit).

To treat the case of wiretap codes for fast fading channels, a similar error prob-
ability is performed [2]. Once the fading is fixed, we deal with a Gaussian channel,
and can use the results of [3]. The fading case is then obtained by averaging over
the fading coefficients. The upper bound on Eve’s probability of correct decision
depends on parameters given by the channel, and by an expression that depends
on the coding strategy, which is the part that can be optimized, leading to a design
criterion: to minimize
\[
\sum_{x \in \Lambda, x \neq 0} \prod_{x_i \neq 0} \frac{1}{|x_i|^3}
\]
where \(x = (x_1, \ldots, x_n)\). A first observation is that it is desirable to have as many
non-zero coefficients among the \(x_i\). Ideally, we would like all the non-zero vec-
tors \(x \in \Lambda\) to have only non-zero coefficients. Given a vector \(x\), its diversity is
the number of its non-zero coefficients. Given a lattice, its diversity is the mini-
imum diversity over every non-zero lattice vector. The property of diversity is well
understood in the context of ideal lattices.

2.2 Ideal Lattices

Let \(K\) be a number field, that is an extension of \(\mathbb{Q}\), of degree \(n\), and let \(\sigma_1, \ldots, \sigma_{r_1}\)
be the \(r_1\) real embeddings of \(K\), and \(\sigma_{r_1+1}, \ldots, \sigma_{r_1+2r_2}\) be the complex embeddings
of \(K\), which come in \(r_2\) pairs of conjugates. We have that \(n = r_1 + 2r_2\). If \(r_2 = 0\), we
say that \(K\) is a totally real number field. For \(x \in K\), recall that the trace of
\(x\) over \(\mathbb{Q}\) is defined as \(\text{Tr}_{K/\mathbb{Q}}(x) = \sum_{i=1}^{n} \sigma_i(x)\), while the norm of \(x\) is defined as
\(N_{K/\mathbb{Q}}(x) = \prod_{i=1}^{n} \sigma_i(x)\).

From now on, we assume that \(K\) is a totally real number field of degree \(n\)
\((n = r_1)\), with ring of integers \(\mathcal{O}_K\), and real embeddings \(\sigma_1, \ldots, \sigma_n\).

**Definition 1** An ideal lattice is an integral lattice \((\mathcal{I}, q_\alpha)\), where \(\mathcal{I}\) is an \(\mathcal{O}_K\)-ideal
(which may be fractional) and

\[ q_\alpha : \mathcal{I} \times \mathcal{I} \to \mathbb{Z}, \quad q_\alpha(x, y) = \text{Tr}_{K/\mathbb{Q}}(\alpha xy), \quad \forall x, y \in \mathcal{I} \]

where \(\alpha \in K\) is totally positive (i.e., \(\sigma_i(\alpha) > 0\) for all \(i\)).

The element \(\alpha\) has a “twisting” effect which is useful to obtain different types
of lattices over the same ring of integers.
If \( \{\omega_1, \ldots, \omega_n\} \) is a \( \mathbb{Z} \)-basis of \( \mathcal{I} \), the generator matrix \( M \) of the corresponding ideal lattice \( (\mathcal{I}, q_\alpha) = \{x = uM | u \in \mathbb{Z}^n\} \) is given by

\[
M = \begin{pmatrix}
\sqrt{\alpha_1} \sigma_1(\omega_1) & \sqrt{\alpha_2} \sigma_2(\omega_1) & \cdots & \sqrt{\alpha_n} \sigma_n(\omega_1) \\
\vdots & \vdots & \ddots & \vdots \\
\sqrt{\alpha_1} \sigma_1(\omega_n) & \sqrt{\alpha_2} \sigma_2(\omega_n) & \cdots & \sqrt{\alpha_n} \sigma_n(\omega_n)
\end{pmatrix}
\]

(4)

where \( \alpha_j = \sigma_j(\alpha) \), for all \( j \). One easily verifies that the Gram matrix \( MM^t \) is given by \( \{\text{Tr}(\alpha \omega_j)\}^{n}_{i,j=1} \).

Ideal lattices are defined over a number field, irrespectively of its signature \((r_1, r_2)\). However, when \( r_2 = 0 \) as assumed above, the diversity of the resulting lattice is maximal, that is \( n = r_1 \). Indeed, if \( x \) is a non-zero vector in this ideal lattice, then

\[
x = \left( \sqrt{\alpha_1} \sigma_1(\sum_{i=1}^n u_i \omega_i), \ldots, \sqrt{\alpha_n} \sigma_n(\sum_{i=1}^n u_i \omega_i) \right),
\]

but if \( x_j = \sqrt{\alpha_i} \sigma_1(\sum_{i=1}^n u_i \omega_i) = 0 \) for some \( j \), then \( \sum_{i=1}^n u_i \omega_i = 0 \) and every \( x_i \) must be zero, a contradiction.

When choosing \( \Lambda_c \) to be an ideal lattice from a totally real number field \( K \) of degree \( n \), (3) becomes

\[
\sum_{x \in \Lambda_c, x \neq 0} \prod_{x \neq 0} \frac{1}{|x_i|^3} = \sum_{x \in \mathcal{I}, x \neq 0} \prod_{i=1}^n \frac{1}{\alpha_i |\sigma_i(x)|^3} = \sum_{x \in \mathcal{I}, x \neq 0} \frac{1}{|\mathcal{N}_{K/Q}(x)|^3} \tag{5}
\]

since \( x = (x_1, \ldots, x_n) = (\sqrt{\alpha_1} \sigma_1(x), \ldots, \sqrt{\alpha_n} \sigma_n(x)) \) for some \( x = \sum_{i=1}^n u_i \omega_i \in \mathcal{I} \). In this paper, we will typically consider the case where \( K \) is a Galois extension, and \( \mathcal{I} \) is a principal, to start with (in whole generality, a Galois extension is not needed to build an ideal lattice, and the ideal does not have to be principal). In that case, \( x \in \mathcal{I} = (\alpha) \mathcal{O}_K, N_{K/Q}(x) = N_{K/Q}(\alpha) N_{K/Q}(x') \) for some \( x' \in \mathcal{O}_K \), and we see from (5) that the sum which becomes of interest is

\[
\sum_{x \in \mathcal{O}_K, x \neq 0} \frac{1}{|\mathcal{N}_{K/Q}(x)|^3}. \tag{6}
\]

It is worth noting that since \( \Lambda_c \) is a sublattice of \( \Lambda_b \), both lattices will be obtained as ideal lattices over \( \mathcal{O}_K \). This is consistent with the design of \( \Lambda_b \), since this lattice will also benefit of the full diversity property coming from choosing \( K \) totally real, and full diversity is indeed a design criterion for reliability over fast Rayleigh fading channels [10].

3 Some Number Fields with Prescribed Ramification

Let us make a first obvious remark regarding (6). This sum will not be finite in most of the cases. Indeed, for it to be finite, we need \( \mathcal{O}_K \) to contain finitely many elements of norm 1, that is finitely many units. It is known (this is Dirichlet’s unit Theorem) that the group of units of \( K \) is finitely generated and has rank \( r_1 + r_2 - 1 \),
as a result, the rank will always be strictly positive, except for quadratic imaginary fields (for which \( r_1 = 0 \) and \( r_2 = 1 \)).

The infinite sum (6) where \( x \in \mathcal{O}_K \) comes from computing a probability of error over the whole lattice, instead of considering the actual signal constellation, which is a subset of it. A bound involving the whole lattice provides an upper bound on a finite constellation, and is often easier to handle, however, in this case, it is meaningful only for quadratic imaginary fields, and in general, the bound will have to involve a finite set of points:

\[
\sum_{x \in \mathcal{O}_K \cap \mathcal{R}, x \neq 0} \frac{1}{|N_{K/Q}(x)|^3},
\]

where \( \mathcal{R} \) decides of the shape of the finite constellation.

### 3.1 Norms and Ramification

The dominant terms in this sum are integers with small norms. The contribution of the terms of norm 1 depends on the density of units, which we will discuss in Section 4, while that of elements of norm at least 2 depends on the ramification in \( K \), the class number of \( K \), and also the density of units (since if \( x \) has a given norm, then \( ux \) where \( u \) is a unit will have the same norm).

Since the examples we will consider all have a class number of 1, and the density of units will be discussed in the next section, we next study the ramification effect.

Let \( p \) be a prime, then \( p \) belongs to the ideal \( p\mathcal{O}_K \), and by considering the prime factorization of \( p\mathcal{O}_K \), we have

\[
N(p\mathcal{O}_K) = N(\prod_{i=1}^{g} p_i^{e_i}) = \prod_{i=1}^{g} N(p_i)^{e_i} = |N_{K/Q}(p)| = p^n
\]

where all \( p_i \) are distinct prime ideals. If \( K \) is Galois, then \( p\mathcal{O}_K = \prod_{i=1}^{g} p_i^{e_i} \), that is \( e_i = e \) for all \( i \).

In particular, if \( p \) is totally ramified (\( g = 1 \) and \( e_1 = n \)) or if \( p \) totally splits (\( g = n \) and all the ramification indices are 1), then

\[
N(p)^n = p^n, \text{ or } \prod_{i=1}^{n} N(p_i) = p^n
\]

shows the existence of an ideal above \( p \) of norm \( p \). If this ideal is principal, then the generators will have norm \( p \). This argument shows how to find elements of norm \( p \), but also that having \( e = g = 1 \) will force the smallest norm involving only the prime \( p \) to be at least \( p^n \). Indeed, suppose that there exists an element \( x \in \mathcal{O}_K \) whose norm is \( p^j \) for some positive \( j \). Then \( N((x)\mathcal{O}_K) = |N_{K/Q}(x)| = p^j \) and by definition \( |\mathcal{O}_K/(x)\mathcal{O}_K| = p^j \), which shows that \( p^j \subset (x)\mathcal{O}_K \), thus \( (p^j)\mathcal{O}_K \subset (x)\mathcal{O}_K \) and \( (x)\mathcal{O}_K \subset (p^j)\mathcal{O}_K \). Since \( (p)\mathcal{O}_k \) is prime (\( e = g = 1 \)), it must be that \( (x)\mathcal{O}_K = (p^j)^{j'} \mathcal{O}_K \) for some \( j' \leq j \), and \( N((x)\mathcal{O}_K) = N((p)\mathcal{O}_K)^{j'} = p^{nj'} \), showing that \( j \geq n \).
3.2 Maximal Real Subfields of Cyclotomic Fields

Let $\zeta_p$ denote a primitive $p$th root of unity, and consider the cyclotomic field $\mathbb{Q}(\zeta_p)$. It has degree $p - 1$ over $\mathbb{Q}$, and its maximal real subfield $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$ has degree $(p - 1)/2$ over $\mathbb{Q}$. They have respective rings of integers $\mathbb{Z}[\zeta_p]$ and $\mathbb{Z}[\zeta_p + \zeta_p^{-1}]$. The ramification in $\mathbb{Q}(\zeta_p)$ is well understood.

**Theorem 1** [9] Let $q$ be a rational prime different from $p$, then $q$ is unramified in $\mathbb{Q}(\zeta_p)$ and in fact

$$ (q)\mathbb{Z}[\zeta_p] = q_1 \cdots q_g $$

with mutually distinct prime ideals $q_i$ and each of inertial degree $f = f(q_i/q)$ equal to the order of $q$ in $(\mathbb{Z}/p\mathbb{Z})^\times$, i.e., $f$ is the least natural number such that

$$ q^f \equiv 1 \pmod p. $$

Since we will be looking at subfields of $\mathbb{Q}(\zeta_p)$, it is useful to remember that the ramification and the residual index satisfy transitivity, namely

$$ e(q_{KL}/q) = e(q'/q) e(q_{KL}/q') \quad \text{and} \quad f(q_{KL}/q) = f(q/\mathbb{Q}) f(q_{KL}/q), \quad (8) $$

for the tower $L/K/\mathbb{Q}$ and $q_{KL}$ a prime above $q_K$, and $q_K$ a prime above $q$.

Consider the special case when $p = 2p' + 1$, with $p'$ a prime. It is then easy to make sure that small primes stay inert in $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$.

**Lemma 1** Suppose that $p = 2p' + 1$, where both $p$ and $p'$ are prime (such a prime $p'$ is called a Sophie Germain prime). Then the primes smaller than $p$ are inert in $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$.

**Proof** Let $q$ be a prime smaller than $p$. By forcing $p = 2p' + 1$ with $p'$ prime, the degree of $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$ is now $p'$ over $\mathbb{Q}$. Since

$$ p' = e(q_i/q) f(q_i/q) g(q_i/q) $$

for every prime $q_i$ above $q$ and $e(q_i/q) = 1$ when $q$ is distinct from $p$ (if $e(q_i/q) > 1$, then by transitivity $(8)$ $q$ should ramify in $\mathbb{Q}(\zeta_p)$), we deduce that either (1) $f(q_i/q) = 1$ and $g(q_i/q) = p'$, or (2) $f(q_i/q) = p'$ and $g(q_i/q) = 1$. But $f(q_i/q) = 1$ implies that either $q \equiv 1 \pmod p$, or $q^2 \equiv 1 \pmod p$.

The former case cannot happen if $q$ is smaller than $p$.

The latter case is also impossible. Here are two reasons why this is the case. Firstly: $q^2 \equiv 1 \pmod p$ means that $q$ is an element of order 2. But in the cyclic group $(\mathbb{Z}/p\mathbb{Z})^\times$, generated by some element $a$, the elements of order 2 are of the form $a^k$ with $(p - 1)/\gcd(p - 1, k) = 2$, that is $2p'/\gcd(2p', k) = 2$, implying that $k$ must be an odd multiple of $p'$, that is $k = p'$. Now this element of order 2 has to be $a^{p'} = p_1$, since $(p - 1)^2 \equiv 1 \pmod p$. Alternatively $1$: $q^2 \equiv 1 \pmod p$ is equivalent to $(q - 1)(q + 1) \equiv 0 \pmod p$, that is $p$ divides $(q - 1)$ or $(q + 1)$.

But $q < p$ so $p$ cannot divide $(q - 1)$, and $p$ cannot divide $q + 1$ either, since $p - 1$ is even.

This shows that $f(q_i/q) = p'$ and $q$ is inert.

\textsuperscript{1} We would like to thank the anonymous reviewer who proposed this simplification.
Example 1 Consider \( \mathbb{Q}(\zeta_{23}) \), with \( 23 = 2 \cdot 11 + 1 \). The primes 2, 3, 5, 7, 11, 13, 17, 19 are all inert in \( \mathbb{Q}(\zeta_{23} + \zeta_{23}^{-1}) \).

We could consider more generally a subfield \( K \) of \( \mathbb{Q}(\zeta_p) \) of degree \( [K : \mathbb{Q}] = m \), when \( p = mp' + 1 \). Indeed, the Galois group of \( \mathbb{Q}(\zeta_p) / \mathbb{Q} \) is isomorphic to \( (\mathbb{Z}/p\mathbb{Z})^\times \), that is, it is a cyclic group of order \( p - 1 = mp' \). Let us denote by \( \sigma \) its generator. Then \( \sigma^{p'} \) generates a subgroup of order \( m \), to which corresponds a subfield \( K = \mathbb{Q}(\zeta_p)_{\langle \sigma^{p'} \rangle} \) which is fixed by \( \langle \sigma^{p'} \rangle \), which is of degree \( p' \) over \( \mathbb{Q} \). Since \( p' \) is prime, the same argument as in the proof of Lemma 1 shows that if \( q \) is a prime different from \( p \), then \( q \) cannot ramify. Either it is inert, or it splits totally.

Example 2 Consider \( \mathbb{Q}(\zeta_{67}) \), with \( 67 = 6 \cdot 11 + 1 \). Let \( \sigma \) be the generator of the Galois group of \( \mathbb{Q}(\zeta_{67}) / \mathbb{Q} \). The subgroup \( \langle \sigma_{11} \rangle \) has order 6, with corresponding fixed field \( K \), which is of degree 11 over \( \mathbb{Q} \). Its minimal polynomial is \( x^{11} - x^{10} - 30x^9 + 63x^8 + 220x^7 - 698x^6 - 101x^5 + 1960x^4 - 1758x^3 + 35x^2 + 243x + 29 \). Since \( 11 = f(q_i/q)g(q_i/q) \) we have that \( f(q_i/q) \) is either 1 or 11. Let us assume that this is 1. Using the transitivity formula (8)

\[
f(q_L/q) = f(q_L/q_i)
\]

thus \( f(q_L/q) \) is either 1, 2, 3, or 6, with \( L = \mathbb{Q}(\zeta_{67}) \). A direct computation using Theorem 1 shows that 2, 3, 7, 11, 13, 17, 19 and 23 are inert. On the other hand, \( 29^3 \equiv 1 \pmod{67} \).

Among those totally real fields with inert small primes, we now look at those with small number of units, as considered next.

4 Units and Regulator

Let \( K \) be a number field of degree \( d \) and signature \( (r_1, r_2) \). Set \( r = r_1 + r_2 - 1 \).

The density of units in \( K \) is related to its regulator \( R \).

**Definition 2** Given a basis \( e_1, \ldots, e_r \) for the group of units modulo the group of roots of unity. The **regulator** of \( K \) is

\[
R = | \det(\log |\sigma_i(e_j)|)|_{1 \leq i, j \leq r},
\]

where \( |\sigma_i(e_j)| \) denotes the absolute value for the real embeddings, and the square of the complex absolute value for the complex ones.

Let \( w \) be the number of roots of unity in \( K \). The best known bound on the number of units is given in [4].

**Theorem 2** The number of units \( U(q) \) such that \( \max_{1 \leq i \leq d} |\sigma_i(u)| < q \) in \( K \) is given by

\[
U(q) = \frac{w(r + 1)^r}{Rr!}((log q)^r + O((log q)^{r-1-(cR^{2/r})^{-1}}))
\]

as \( q \to \infty \) and \( c = 6 \cdot 2 \cdot 10^{12}d^{10}(1 + 2 \log d) \).
Table 1 Some totally real number fields $K$ with their small primes and regulator. The first column describes $K$ as a subfield of a cyclotomic field $\mathbb{Q}(\zeta_p)$, the second column is the regulator $R$, the third column is the minimal polynomial of $K$, and the fourth column gives the first small prime which is not inert.

<table>
<thead>
<tr>
<th>$K \subset \mathbb{Q}(\zeta_p)$</th>
<th>$R$</th>
<th>$p(X)$</th>
<th>primes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Q}(\zeta_{11})$</td>
<td>1.63</td>
<td>$x^5 + x^4 - 4x^3 - 3x^2 + 3x + 1$</td>
<td>11 ramifies</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{13})$</td>
<td>30.36</td>
<td>$x^5 - 9x^4 + 20x^3 - 5x^2 - 11x - 1$</td>
<td>5 splits</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{23})$</td>
<td>123.32</td>
<td>$x^5 - x^4 - 16x^3 - 5x^2 + 21x + 9$</td>
<td>3 splits</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{31})$</td>
<td>1014.31</td>
<td>$x^{11} + x^{10} - 10x^9 - 9x^8 + 36x^7 + 28x^6$</td>
<td></td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{41})$</td>
<td>330512.24</td>
<td>$x^{11} - x^{10} - 30x^9 + 63x^8 + 220x^7 - 698x^6$</td>
<td>23 ramifies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-101x^5 + 1960x^4 - 1758x^3 + 35x^2 + 243x + 29$</td>
<td></td>
</tr>
</tbody>
</table>

By choosing $R$ accordingly, we might use this result on $U(q)$ to evaluate the amount of units, that is of elements of norm 1 in (7). Since we focus on totally real number fields, $w = 2$ (the only roots of unity are $\pm 1$), $r_2 = 0$ and $r_1 = d$, so that $r = d - 1$. Thus the regulator is the only factor that distinguishes two totally real numbers of same degree.

Unfortunately, results on regulators of number fields are not easily found. To have a sense of the range to which regulators belong, we compute numerically regulators corresponding to totally real number fields identified earlier. Note that the numerical computations are not obvious either, since they require units computations, which are lengthy, when the degree of the number fields increases. Examples of number fields can be found in Table 1. We observe that though maximal real subfields have the advantage of having small primes under control, they also have very small regulators.

The case of degree 5 shows that the choice of the regulator is making a huge difference, since the dominant term for $U(q)$ is

$$
\frac{2 \cdot 5^4}{4!R}(\log q)^4 = \frac{625}{12R}(\log q)^4
$$

yielding respectively

$$
\sim 0.4(\log q)^4, \quad \sim 32(\log q)^4
$$

for the smallest and biggest regulators shown in Table 1.

In fact, this difference is so big that it might make irrelevant the consideration about the ramification of small primes. The picture for wiretap lattice codes is however more complex, since it involves the design of not only $\Lambda_e$ (the lattice designed to confuse Eve), but also $\Lambda_b$ (the lattice that provides reliability for Bob). Though this has not been made completely explicit in this particular context of coset encoding, the discriminant $d_K$ of the number field $K$ is known [5, 1] to play a role in the design of $\Lambda_b$, and in fact, it is usually preferred to be not too big. Table 2 illustrates how the discriminant $d_K$ grows with the regulator. This suggests that the optimal design might be a trade-off between the discriminant and the regulator, and that further benefits in terms of confusion could be obtained by considering the ramification of small primes.
Table 2 Some totally real number fields $K$ with their small primes, discriminant $d_K$, regulator and class number $h_K$. The first column describes $K$ as a subfield of a cyclotomic field $\mathbb{Q}(\zeta_p)$, the second column is the regulator $R$, the third column is discriminant $d_K$ of $K$, and the fourth column gives the first small prime which is not inert.

<table>
<thead>
<tr>
<th>$K \subset \mathbb{Q}(\zeta_p)$</th>
<th>$R$</th>
<th>$d_K$</th>
<th>$h_K$</th>
<th>primes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Q}(\zeta_{11})$</td>
<td>1.63</td>
<td>$11^4$</td>
<td>1</td>
<td>11 ramifies</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{31})$</td>
<td>30.36</td>
<td>$31^4$</td>
<td>1</td>
<td>5 splits</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{41})$</td>
<td>123.32</td>
<td>$41^4$</td>
<td>1</td>
<td>3 splits</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{23})$</td>
<td>1014.31</td>
<td>$23^{10}$</td>
<td>1</td>
<td>23 ramifies</td>
</tr>
<tr>
<td>$\mathbb{Q}(\zeta_{67})$</td>
<td>330512.24</td>
<td>$67^{10}$</td>
<td>1</td>
<td>29 splits</td>
</tr>
</tbody>
</table>

5 Future Work

Identifying totally real number fields with prescribed ramification and regulator is only one step in the design of wiretap codes for fast fading channels, and the goal of this paper was to propose different number fields that could be used for that purpose, and to illustrate the trade-offs among the different parameters involved, taking into account solely the design of $\Lambda_e$, the lattice that provides confusion to the eavesdropper. Apart looking for more such fields inside cyclotomic fields with $\mathbb{Q}(\zeta_n)$, $n$ not prime, and possibly among non-abelian Galois extensions, other criteria need to be understood:

- The design of wiretap codes involves the choice of both $\Lambda_e$ and $\Lambda_b$, with $\Lambda_e \subset \Lambda_b$. It is needed to understand which ideal lattices can be built over the number fields identified. For example, it is known that the integer lattice $\mathbb{Z}^{(p-1)/2}$ can be obtained on the ring of integers of the field $\mathbb{Q}(\zeta_p + \zeta_p^{-1})$ with a suitable choice of $\alpha$ [10].

- The design of good lattices to ensure Bob’s reliability needs to be taken into account (see [5] for a related work). In that case, the picture is more complicated, and is likely to involve the discriminant of the number field as well, yielding a trade-off between discriminant and regulator.

- Coset encoding in the context of ideal lattices needs to be addressed, in order to provide an efficient way of encoding (by performing a bit labeling of the lattice points).

These aforementioned items are different pieces of a puzzle, which need to be put together to obtain a fully working wiretap code, whose perform could be simulated.

Acknowledgements We would like to thank the anonymous reviewers for the useful comments.

References

Probability Bounds for Two-Dimensional Algebraic Lattice Codes

David A. Karpuk, Member, IEEE · Camilla Hollanti, Member, IEEE · Emanuele Viterbo, Fellow, IEEE

Abstract In this paper we provide easily computable probability bounds for the correct decision for an eavesdropper, over a fast fading channel, when coset coding using two-dimensional algebraic lattices is employed. These bounds are stated in terms of the group of units and Dedekind zeta function of the underlying number field.

Keywords fading channel · wiretap channel · number fields · lattices · unit group · Dedekind zeta function

1 Introduction

Lattices and lattice codes provide an efficient and robust means for many applications in wireless communications [11]. In particular, number field lattice codes have been applied to Gaussian and fading wiretap channels [7,9,10,4,3], where number theoretic invariants allow one to perform analysis using algebraic data. In addition, lattice codes from totally real number fields exhibit full diversity.

In [7], the authors estimate the probability of the correct decision of an eavesdropper, for lattice codes coming from number fields whose ring of integers is a principal ideal domain (PID). We improve on and generalize these

The research of D. Karpuk and C. Hollanti is supported by the Emil Aaltonen Foundation’s Young Researcher’s Project, and by the Academy of Finland grant #131745.

Part of this work was performed at the Monash Software Defined Telecommunications Lab and was supported by the Monash Professional Fellowship and the Australian Research Council under Discovery grants ARC DP 130100103.

D. Karpuk and C. Hollanti are with the Department of Mathematics and System Analysis, P.O. Box 11100, FI-00076 Aalto University, Finland. E-mail: davekarpuk@aalto.fi, E-mail: camilla.hollanti@aalto.fi · E. Viterbo is with the Department of Electrical and Computer System Engineering, Monash University, Clayton, VIC 3800, Australia. E-mail: emanuele.viterbo@monash.edu
results for two-dimensional number field lattice codes. In particular, we provide easily computable estimates for this probability for lattices coming from arbitrary degree two number fields, and compare these new estimates with those of [7], where the new estimates are shown to perform better.

Similar purely number theoretic work has also been carried out in [6] and [5] on estimating the number of algebraic integers of bounded height. In the realm of information theory, related work has been done by Vehkalahti and Lu in [12–14], where a connection is demonstrated between the unit group and the diversity-multiplexing gain trade-off of division algebra-based space-time codes via inverse determinant sums. Subsequent work in [15] explores this relation further, showing that the density of unit group completely determines the growth of the inverse determinant sum. Our estimates are similarly based on the growth of the unit group, but our goal is to make such estimates as explicit as possible, while providing theoretical tools which can ideally be used to generalize these results to higher dimensions.

Lastly, we provide asymptotic analysis of the estimates of [7], and show that the estimates remain accurate as the size of the underlying constellation increases. Numerical data is provided for examples of number field lattice codes coming from number fields whose ring of integers is not a PID, for which the results of [7] do not apply. The reader can see the performance of our new bounds explicitly from the experimental results.

2 Lattices and Coset Coding

A lattice \( \Lambda \) is a discrete abelian subgroup of a real vector space, and can be described as the set \( M \cdot \mathbb{Z}^n \), where \( M \in M_n(\mathbb{R}) \) is the generator matrix for the lattice. The rank of the lattice is the rank of the matrix \( M \). Lattices and lattice codes are ubiquitous in the study of fading channels; see [11].

In a wiretap channel, Alice is transmitting confidential data to the intended receiver Bob over a fading channel, while an eavesdropper Eve tries to intercept the data received over another fading channel. Following [2], we model this channel by the equations

\[
y_b = H_b x + z_b, \quad y_e = H_e x + z_e
\]

where \( y_b \in \mathbb{R}^n \) is Bob’s received vector, and \( z_b \) is Bob’s noise vector, assumed to be zero-mean Gaussian with variance \( \sigma_b^2 \). Bob’s channel matrix is of the form \( H_b = \text{diag}(|h_{b,i}|) \) for \( h_{b,i} \in \mathbb{C} \) whose entries \( h_{b,i} \) are complex zero-mean Gaussian random variables, so that the non-zero entries of \( H_b \) are Rayleigh distributed fading coefficients. The quantities in the second equation are the corresponding variables for Eve’s channel. The intended signal is \( x \in \mathbb{R}^n \).

We assume that both Bob and Eve have perfect channel state information, while Alice has none. The security of the fading wiretap channel is based on the assumption that Bob’s SNR is sufficiently large compared to Eve’s SNR, that is, that \( \sigma_b^2 \gg \sigma_e^2 \). Alice exploits these physical conditions by using a coset coding strategy [16] in order to confuse Eve. Alice’s coding strategy begins by
choosing a lattice $A_b$ containing information symbols intended for Bob, and a sublattice $A_e$ containing random information intended to confuse Eve. Alice’s codebook consists cosets of $A_e$ in $A_b$. To transmit, Alice chooses a codeword $c$ and a random vector $r \in A_e$ and then sends the signal

$$x = c + r \in c + A_e,$$

(2)

where $c$ contains the data bits and $r$ embeds the random bits. Eve’s SNR is assumed sufficiently large so that she can decode $r$ perfectly. However, the codewords $c$ belong to a finer lattice, and thus it is much more unlikely that Eve can correctly decode $c$.

By [2] the expression $\bar{P}_{c,e}$ for the average probability of Eve correctly decoding $c$ over a fast fading channel is approximately

$$\bar{P}_{c,e} \simeq \left( \frac{1}{4\gamma_e^2} \right)^{n/2} \frac{\text{Vol}(A_b)}{\sum_{x \in A_e} \prod_{i=1}^{n} \frac{1}{|x_i|^3}},$$

(3)

where $\gamma_e$ is Eve’s average SNR over the $n$ channels. Our goal is to approximate $\sum_{x \in A_e} \prod_{i=1}^{n} \frac{1}{|x_i|^3}$ for algebraic lattices when $n = 2$.

### 3 Number Theoretic Background

We assume the reader is familiar with basic algebraic structures such as fields, rings, and ideals. The definitions presented here generalize to arbitrary number fields, but for the purpose of eliminating unnecessary theoretical machinery we have simplified them for real quadratic number fields. We recommend [8] as a blanket reference for all necessary number theoretic results.

A real quadratic number field is a field $K = \mathbb{Q}(\sqrt{d}) = \{a + b\sqrt{d} : a, b \in \mathbb{Q}\}$ for a square-free positive integer $d$. The ring of integers $\mathcal{O}_K$ of $K$ is the set $\{a + b\omega : a, b \in \mathbb{Z}\}$, where

$$\omega = \begin{cases} 
(1 + \sqrt{d})/2 & \text{if } d \equiv 1 \pmod{4} \\
\sqrt{d} & \text{otherwise.}
\end{cases}$$

(4)

One can easily verify that $\mathcal{O}_K$ has rank 2 as an abelian group. The discriminant $D_K$ of $K$ is

$$D = D_K = \begin{cases} 
d & \text{if } d \equiv 1 \pmod{4} \\
4d & \text{otherwise.}
\end{cases}$$

(5)

Given an ideal $a$ of $\mathcal{O}_K$, it can be factored uniquely into prime ideals $p_1$, in much the same way one factors integers: $a = p_1^{e_1} \cdots p_g^{e_g}$. The ideal $(p)$ of $\mathcal{O}_K$ may no longer be prime, but it can be factored into prime ideals, and the possibilities for the factorization of $(p)$ in $\mathcal{O}_K$ are:

$$(p) \equiv \begin{cases} 
(p) \text{ is prime} & \text{iff } (p, D) = 1, D \not\equiv y^2 \pmod{p}, \text{ for any } y \in \mathbb{Z} \\
p \mid D & \text{iff } (p, D) = 1, D \equiv y^2 \pmod{p}, \text{ for some } y \in \mathbb{Z} \\
pq & \text{iff } p \mid D.
\end{cases}$$

(6)
In the first case we say that \( p \) is inert in \( K \), in the second that \( p \) splits in \( K \), and in the third that \( p \) ramifies in \( K \). If a prime ideal \( \mathfrak{p} \) of \( \mathcal{O}_K \) appears in the factorization of \( (p) \), we say that \( p \) divides \( p \), which we denote by \( \mathfrak{p} \mid p \).

**Example 1** Let \( K = \mathbb{Q}(\sqrt{10}) \). The discriminant of \( K \) is \( D = 40 \), thus the primes that ramify in \( K \) are \( p = 2, 5 \). In fact, we have the prime factorizations

\[
(2) = p_2^2, \quad p_2 = (2, \sqrt{10}), \quad \text{and} \quad \tag{7}
\]

\[
(5) = p_5^2, \quad p_5 = (5, \sqrt{10}). \quad \tag{8}
\]

If \( p = 13 \) we have \( 40 \equiv 1^2 \pmod{13} \), hence the prime 13 splits in \( \mathcal{O}_K \):

\[
(13) = \mathfrak{p}_1 \mathfrak{p}_2, \quad \mathfrak{p}_1 = (13, 6 + \sqrt{10}), \quad \mathfrak{p}_2 = (13, 6 - \sqrt{10}). \quad \tag{9}
\]

We can describe the structure of the group of units \( \mathcal{O}_K^\times \) of the ring \( \mathcal{O}_K^\times \) explicitly as follows. There is a unit \( \epsilon \in \mathcal{O}_K^\times \) such that \( \mathcal{O}_K^\times = \langle \pm \epsilon \rangle \cong \mathbb{Z}_2 \times \mathbb{Z} \), which is called a fundamental unit for \( K \). The fundamental unit can be chosen (up to multiplication by \( \pm 1 \)) such that among all \( u \in \mathcal{O}_K^\times \) with the property that \( |u| > 1, |\epsilon| \) is minimal. The regulator of \( K \) is defined to be \( \rho_K = \log |\epsilon| \).

The regulator of \( K \) measures the density of the units of \( \mathcal{O}_K \) in \( \mathbb{R} \).

Let \( \sigma : K \to K \) be the function \( \sigma(a + b\sqrt{d}) = a - b\sqrt{d} \). We use \( \sigma \) to define the field norm \( N : K \to \mathbb{Q} \), by \( N(x) = x \cdot \sigma(x) \). If \( x \in \mathcal{O}_K \), then \( N(x) \in \mathbb{Z} \).

One can use the field norm to classify units in \( \mathcal{O}_K \), in that for \( x \in \mathcal{O}_K \), we have \( N(x) = \pm 1 \) if and only if \( x \in \mathcal{O}_K^\times \).

**Example 2** Let us continue with the example of \( K = \mathbb{Q}(\sqrt{10}) \). If \( \epsilon = a + b\sqrt{10} \) is a fundamental unit for \( K \), we see by taking norms that \( a \) and \( b \) must satisfy

\[
\pm 1 = \epsilon \cdot \sigma(\epsilon) = (a + b\sqrt{10})(a - b\sqrt{10}) = a^2 - b^210. \quad \tag{10}
\]

From this equation, one can show that \( \epsilon = -3 + \sqrt{10} \) is a fundamental unit for \( K \). Therefore \( \rho_K = \log |\epsilon| = 1.81844645923207... \)

Suppose that \( \mathfrak{p} \) is a prime ideal of \( \mathcal{O}_K \) such that \( \mathfrak{p} \mid p \) for a prime number \( p \in \mathbb{Z} \). The ideal norm of \( \mathfrak{p} \) is

\[
N(\mathfrak{p}) = \begin{cases} 
p^2 & \text{if } \mathfrak{p} \text{ is inert in } K \\
p & \text{otherwise} \end{cases} \quad \tag{11}
\]

One can extend this definition to any ideal \( \mathfrak{a} \) of \( \mathcal{O}_K \) by using prime factorization. If \( \mathfrak{a} = \mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_g^{e_g} \), then

\[
N(\mathfrak{a}) = N(\mathfrak{p}_1)^{e_1} \cdots N(\mathfrak{p}_g)^{e_g} \quad \tag{12}
\]

The two definitions of norm coincide for principal ideals, in the sense that if \( \mathfrak{a} = (x) \) is principal, then \( N(\mathfrak{a}) = |N(x)| \).

The Dedekind zeta function of a number field \( K \) is defined by

\[
\zeta_K(s) = \sum_{\mathfrak{a} \subseteq \mathcal{O}_K} \frac{1}{N(\mathfrak{a})^s} = \sum_{k \geq 1} \frac{a_k}{K^s}, \quad \tag{13}
\]
where $a$ runs over the nonzero ideals of $O_K$, and $a_k$ denotes the number of ideals of $O_K$ of norm $k$. We will only be interested in values of $\zeta_K(s)$ for $s \in \mathbb{R}$, $s > 1$, where classical results guarantee convergence.

The partial Dedekind zeta function of a number field $K$ is defined by

$$\zeta^1_K(s) = \sum_{a' \subseteq O_K} \frac{1}{N(a')^s} = \sum_{k \geq 1} a^1_k k^{-s}$$

(14)

where $a'$ runs over the nonzero principal ideals of $O_K$, and $a^1_k$ denotes the number of principal ideals of $O_K$ of norm $k$. Again, classical results guarantee that this series converges for $s \in \mathbb{R}, s > 1$.

4 Algebraic Lattice Codes

Let $K = \mathbb{Q}(\sqrt{d})$ be a real quadratic field. The canonical embedding $\psi : K \hookrightarrow \mathbb{R}^2$ is defined by $\psi(x) = (x, \sigma(x))$. The set $\psi(O_K)$ is a lattice in $\mathbb{R}^2$, which we call an algebraic lattice. We apply algebraic lattices to wiretap channels by setting Eve’s lattice to be $\Lambda = \Lambda_e = \psi(O_K)$.

We will carve a square constellation centered at the origin of $\mathbb{R}^2$ from the lattice $\Lambda$, by using a bounding box $B$ of side length $2R$ for some $R > 0$. In order to classify algebraic integers in $B$, we define the height of $x \in O_K$ to be

$$H(x) = \max\{|x|, |\sigma(x)|\}.$$  

(15)

Our constellation is then the set $\{\psi(x) \in \Lambda : H(x) \leq R\}$.

When using an algebraic lattice, one sees by (3) that the corresponding probability of Eve’s correct decision is proportional to the inverse norm sum

$$S_{K,R}(s) = \sum_{0 < H(x) \leq R} \frac{1}{|N(x)|^s} = \sum_{k \geq 1} b_{k,R} k^{-s}$$

(16)

when $s = 3$, where we have truncated the inverse norm sum to reflect the fact that we are using a finite constellation. In the above expression $b_{k,R}$ denotes the number of algebraic integers with norm $\pm k$ and height bounded above by $R$. Note that for $k > R^2$, we have $b_{k,R} = 0$ as $|N(x)| = |x \cdot \sigma(x)| \leq (\max\{|x|, |\sigma(x)|\})^2 = H(x)^2$.

5 Lower and Upper Bounds for $S_{K,R}(s)$

We present bounds for $S_{K,R}(s)$ when $s > 1$. These apply to the wiretap channel when $s = 3$, and to the pairwise error probability when $s = 2$ [11]. To obtain a lower bound for the value of $S_{K,R}(s)$, we truncate it after the first term. From (16) we see that $S_{K,R}(s) \geq b_{1,R}$, where $b_{1,R}$ is the number of units of $O_K$ with height bounded above by $R$. As the main contribution to $S_{K,R}(s)$ comes from the units, it is beneficial to have a closed-form expression for $b_{1,R}$.
Fig. 1 On the left, the points of the lattice $\psi(\mathcal{O}_K)$ for $K = \mathbb{Q}(\sqrt{5})$ with a bounding box of side length 10, lying on the hyperbolas $xy = \pm k$. On the right, the same lattice points after applying the transformation $X = \log |x|$, $Y = \log |y|$, lying on the lines $X + Y = \log k$.

To establish an explicit formula for $b_{1,R}$, we apply the 2-to-1 mapping $X = \log |x|$, $Y = \log |y|$ to our lattice $\Lambda$. The image of $\mathcal{O}_K \times \mathbb{K}$ is a 1-dimensional lattice in $\mathbb{R}^2$ living on the line $X + Y = 0$, which we will call $\Lambda_{\log}$ (see Fig. 1). The volume of the lattice $\Lambda_{\log}$ is $\sqrt{2} \rho_K$. We will denote by $\log |B| = (\infty, \log(R)]^2$ the image of the bounding box under the log transformation.

**Proposition 1** For all $s$ we have the lower bound

$$b_{1,R} = 4 \left\lfloor \frac{\log(R)}{\rho_K} \right\rfloor + 2 \leq s_{K,R}(s). \quad (17)$$

**Proof** First, note that $b_{1,R}/2$ is the number of points of $\Lambda_{\log}$ inside $|B|$. By projecting $\Lambda_{\log}$ onto the horizontal axis, one can see that $b_{1,R}/2$ is the number of multiples of $\rho_K$ in the interval $[−\log(R), \log(R)]$. Including the origin, this is equal to $2\lfloor \log(R)/\rho_K \rfloor + 1$, and the result follows. □

**Proposition 2** For all $s$ we have the upper bound

$$b_{1,R} \zeta_{K}^{-1}(s) > s_{K,R}(s). \quad (18)$$

**Proof** We have the trivial bound $b_{K,R} \leq a_k^1 b_{1,R}$, hence

$$s_{K,R}(s) = \sum_{k \geq 1} b_{K,R} \frac{1}{k^s} \leq b_{1,R} \sum_{k \geq 1} a_k^1 \frac{1}{k^s} = b_{1,R} \zeta_{K}^{-1}(s). \quad (19)$$

Note that the only term in $b_{1,R}$ and $b_{1,R} \zeta_{K}^{-1}(s)$ that depends on $R$ is $\log(R)$, thus it is easy to measure how these bounds vary as the size of the constellation changes. In [7], bounds on $s_{K,R}(s)$ were presented in terms of bounded-height Dedekind zeta functions for number fields whose rings of integers are PIDs. However, the above bounds apply to all real quadratic number fields.
6 Approximating the Inverse Norm Power Sum

In this section we recall the method of [7] for approximating the sum $S_{K,R}(s)$. Two elements of $O_K$ generate the same principal ideal if and only if they differ multiplicatively by a unit. It follows that the intersection of $\log|\psi(O_K)|$ and the line $X + Y = \log k$ is the union of $a_k^1$ translates of $A_{\log}$. We also have

$$\text{vol}(\log |B| \cap (X + Y = \log k)) = \sqrt{2} \log(R^2/k)$$  \hspace{1cm} (20)

Following [7], we arrive at the approximation of $b_{k,R}$ by a quantity $n_{k,R}$ obtained by dividing this volume by the volume of $A_{\log}$ (and multiplying by $2\alpha_k^1$), as well as an approximation of the inverse norm sum by a quantity $N_{K,R}(s)$:

$$b_{k,R} \approx n_{k,R} := 2\alpha_k^1 \frac{\log(R^2/k)}{\rho_K}, \quad S_{K,R}(s) \approx N_{K,R}(s) := \sum_{k=1}^{R^2} \frac{n_{k,R}}{k^s}. \hspace{1cm} (21)$$

The main difficulty in computing $n_{k,R}$ lies in calculating $a_k^1$. Here we present an algorithm for listing all ideals of $O_K$ which have norm $k$, from which one can compute $a_k^1$ by using a known algorithm to check whether these ideals are principal (such algorithms are implemented in SAGE [1]).

Fix an integer $k > 1$ and factor it into primes as $k = p_1^{e_1} \cdots p_n^{e_n}$. Suppose that $a \subset O_K$ is an ideal such that $N(a) = k$, with unique factorization $a = p_1^{i_1} \cdots p_m^{i_m}$ into prime ideals. Taking norms, we have

$$p_1^{e_1} \cdots p_n^{e_n} = k = N(p_1)^{i_1} \cdots N(p_m)^{i_m}. \hspace{1cm} (22)$$

By unique factorization of integers, we see that every $p$ appearing in the factorization of $a$ divides some $p$ appearing in the factorization of $k$.

Suppose that $p | k$ and that $p$ is inert in $K$, so that for $p|p$ we have $N(p) = p^2$. For $a$ to exist, we must then have that $p$ divides $k$ an even number of times. If this condition holds, then $p = (p)$ is obviously principal, hence it can be ignored in determining whether or not $a$ is principal.

Let us assume that all inert primes dividing $k$ do so an even number of times. We now make a list of all possible ideals of $O_K$ which have norm $k$, by selecting from each split prime $p_{a_i}$ (counting multiplicities) one of the two prime ideals $p_{a_i}$, dividing it, and for each ramified prime $p_{r_j}$ (counting multiplicities), the unique prime ideal $p_{r_j}$ dividing it. We then compute whether or not the ideal $p_{a_1} \cdots p_{a_i} p_{r_1} \cdots p_{r_m}$ is principal. The total number of principal ideals we obtain this way is $a_k^1$.

Example 3 Let $K = Q(\sqrt{229})$, and let $k = 225 = 3^2 \cdot 5^2$. In $K$ the ideals (3) and (5) both split, and we have factorizations

$$3 = p_3 q_3, \quad p_3 = \left(3, \frac{1 - \sqrt{229}}{2}\right), \quad q_3 = \left(3, \frac{1 + \sqrt{229}}{2}\right) \hspace{1cm} (23)$$

$$5 = p_5 q_5, \quad p_5 = \left(5, \frac{7 - \sqrt{229}}{2}\right), \quad q_5 = \left(5, \frac{7 + \sqrt{229}}{2}\right) \hspace{1cm} (24)$$
thus the list of all ideals of norm $k$ is
\[
{\mathfrak p}_3^2{\mathfrak q}_5^2, \, {\mathfrak p}_3{\mathfrak q}_3{\mathfrak p}_5, \, {\mathfrak q}_3^2{\mathfrak p}_5, \, {\mathfrak p}_3^3{\mathfrak p}_5{\mathfrak q}_5, \, {\mathfrak q}_3^3{\mathfrak p}_5{\mathfrak q}_5, \, {\mathfrak p}_3^2{\mathfrak q}_5^2, \, {\mathfrak p}_3{\mathfrak q}_3{\mathfrak p}_5^2, \, {\mathfrak q}_3 q_5^2. \tag{25}
\]
Exactly three of these ideals are principal, so that $a_{225}^1 = 3$. Specifically,
\[
{\mathfrak p}_3^2{\mathfrak q}_5^2 = (2 - \sqrt{229}), \, {\mathfrak p}_3 q_3^2 = (2 + \sqrt{229}), \, {\mathfrak p}_3 q_3 q_5 = (15). \tag{26}
\]

7 Bounding the Error and Asymptotic Behavior

In this section, we bound the absolute error of the estimate of the previous section, by providing a bound $|b_{k,R} - n_{k,R}|$ for all $k$. We then combine these estimates to provide a bound on $|S_{k,R}(s) - N_{K,R}(s)|$ independent of $R$.

**Proposition 3** We have the bound $|b_{k,R} - n_{k,R}| < 2a_k^1$ for all $k$ and all $R$.

**Proof** Let $L_k$ be the intersection of the line $X + Y = \log k$ and the image of the bounding box under the log transformation. Then we can find $\varepsilon$ such that $0 < \varepsilon < \text{vol}(A_{\log})$ and
\[
\text{vol} (L_k) \leq (b_{k,R}/2 - a_k^1) \text{vol}(A_{\log}) + 2a_k^1 \varepsilon. \tag{27}
\]
That is, $L_k$ can be written of a union of at most $(b_{k,R} - a_k^1)$ translates of the fundamental region of $A_{\log}$, and $2a_k^1$ smaller regions which appear at the boundary. Therefore
\[
n_{k,R} = 2a_k^1 \frac{\text{vol} (L_k)}{\text{vol}(A_{\log})} \leq 2 \left( \frac{(b_{k,R}/2 - a_k^1) \text{vol}(A_{\log}) + 2a_k^1 \varepsilon}{\text{vol}(A_{\log})} \right) \tag{28}
\]
\[
= b_{k,R} - 2a_k^1 + 4a_k^1 \varepsilon/\text{vol}(A_{\log}) < b_{k,R} + 2a_k^1 \tag{29}
\]
which implies that $n_{k,R} - b_{k,R} < 2a_k^1$. Now, note that
\[
\frac{b_{k,R}}{2} \leq a_k^1 \left( \frac{\text{vol} (L_k)}{\text{vol}(A_{\log})} + 1 \right) = \frac{n_{k,R}}{2} + a_k^1, \tag{30}
\]
which one can see by counting how many lattice points can fit inside $L_k$. Therefore $b_{k,R} - n_{k,R} < 2a_k^1$, which implies that $|b_{k,R} - n_{k,R}| < 2a_k^1$. □

**Proposition 4** We have $|S_{k,R}(s) - N_{K,R}(s)| < 2\zeta_k^1(s)$.

**Proof** Using the previous proposition, we have
\[
|S_{k,R}(s) - N_{K,R}(s)| = \left| \sum_{k=1}^{R^2} \frac{b_{k,R}}{k^s} - \sum_{k=1}^{R^2} \frac{n_{k,R}}{k^s} \right| \leq \sum_{k=1}^{R^2} \frac{|b_{k,R} - n_{k,R}|}{k^s} \tag{31}
\]
\[
< 2 \sum_{k=1}^{R^2} \frac{a_k^1}{k^s} < 2 \sum_{k=1}^{\infty} \frac{a_k^1}{k^s} = 2\zeta_k^1(s). \tag{32}
\]
As the quantity $\zeta^1_K(s)$ is independent of $R$, we see that
\[
\frac{|S_{K,R}(s) - N_{K,R}(s)|}{S_{K,R}(s)} < \frac{2\zeta^1_K(s)}{S_{K,R}(s)} \to 0
\] (33)
as $R \to \infty$. That is, the relative error of the approximation of [7] goes to zero as the size of the bounding box (i.e. constellation) increases.

From the bounds $b_{1,R} \leq S_{K,R}(s) \leq b_{1,R}\zeta^1_K(s)$ one can conclude that
\[
S_{K,R}(s) \sim \frac{C}{\rho_K} \log(R) + O(1)
\] (34)
as $R \to \infty$, where $C$ is an absolute constant, and the $O(1)$ term depends only on $K$ and $s$. In [14], it is shown that inverse determinant sums exhibit similar behavior, in that they are bounded above and below by terms of the form $C\log(R)$. However, we have made our constants explicit.

8 Experimental Results

We now present examples to demonstrate the accuracy of the bounds $b_{1,R}$ and $b_{1,R}\zeta^1_K(s)$, and the approximation $N_{K,R}(s)$ for $s = 3$ (the wiretap case), for the fields $\mathbb{Q}(\sqrt{5})$, $\mathbb{Q}(\sqrt{10})$, and $\mathbb{Q}(\sqrt{229})$. As we have seen, the second and third fields both have the property that their rings of integers are not PIDs.

\[
\begin{array}{cccc}
\log(R) & b_{1,R} & S_{K,R}(3) & b_{1,R}\zeta^1_K(3) \\
1 & 10 & 10.047250 & 10.275477 \\
2 & 26 & 18.257549 & 18.405860 \\
3 & 34 & 26.480869 & 26.716242 \\
4 & 42 & 34.706803 & 34.936625 \\
5 & 50 & 42.927600 & 43.157007 \\
6 & 58 & 51.210616 & 51.377589 \\
\end{array}
\]

$K = \mathbb{Q}(\sqrt{5})$ :

\[
\begin{array}{cccc}
\log(R) & b_{1,R} & S_{K,R}(3) & b_{1,R}\zeta^1_K(3) \\
1 & 1 & 2.031250 & 2.063041 \\
2 & 2 & 6.079292 & 6.189124 \\
3 & 6 & 6.160755 & 6.189124 \\
4 & 10 & 10.206627 & 10.315207 \\
5 & 10 & 10.287114 & 10.315207 \\
6 & 14 & 14.334457 & 14.441289 \\
\end{array}
\]

$K = \mathbb{Q}(\sqrt{10})$ :

\[
\begin{array}{cccc}
\log(R) & b_{1,R} & S_{K,R}(3) & b_{1,R}\zeta^1_K(3) \\
1 & 1 & 2.031250 & 2.063041 \\
2 & 2 & 2.034669 & 2.063481 \\
3 & 6 & 6.037611 & 6.109443 \\
4 & 10 & 10.106103 & 10.109443 \\
5 & 10 & 10.106103 & 10.109443 \\
6 & 10 & 10.110588 & 10.182405 \\
\end{array}
\]

$K = \mathbb{Q}(\sqrt{229})$ :
If \( u \in \mathcal{O}_K^\times \) and \( u \neq \pm 1 \), then the four distinct elements \( \pm u, \pm \sigma(u) \) all have the same height. As the main contribution to the sum \( S_{K,R}(s) \) comes from the group \( \mathcal{O}_K^\times \), this explains why the value of \( S_{K,R}(s) \) “jumps” by approximately 4 as \( R \) increases.

9 Conclusions and Future Work

We have provided probability bounds for the correct decision of an eavesdropper over a fast fading channel, when two-dimensional number field lattice codes are used. We have shown via experiment that these bounds outperform the previous estimate of [7]. Furthermore, we have generalized this approximation to arbitrary two-dimensional algebraic lattice codes, by providing an algorithm for calculating the number of principal ideals of a fixed norm. Lastly, we have provided asymptotic analysis of the estimate of the inverse norm sum presented in [7].

Future work will involve generalizing the calculation of \( a_1^k \) and providing analogous estimates for higher-dimensional totally real number fields. We also hope to generalize these results to fading MIMO wiretap channels, where a similar analysis of the group of units in a central simple algebra will be necessary.

References

15. Vehkalahti, R., Luzzi, L.: Connecting dmt of division algebra space-time codes and point
Abstract:
The problem of designing physical-layer network coding schemes via lattice partitions is considered. Building on the compute-and-forward relaying strategy of Nazer and Gastpar, who demonstrated its asymptotic gain using information-theoretic tools, an algebraic approach is taken to show its potential in non-asymptotic settings. The resulting scheme naturally leads to a linear network coding channel over modules. In the practically important case where the underlying ring is a finite chain ring, a matrix channel is obtained for which tight capacity bounds and polynomial-complexity capacity-achieving coding schemes can be obtained.

(Joint work with Chen Feng, Danilo Silva, and Roberto W. Nobrega)
On the Rank of Incidence Matrices in Projective Hjelmslev Spaces

Ivan Landjev · Peter Vandendriesche

the date of receipt and acceptance should be inserted later

Abstract Let \( \sigma = (\sigma_1, \ldots, \sigma_n) \) and \( \tau = (\tau_1, \ldots, \tau_n) \) be non-increasing sequences of non-negative integers with \( \sigma \leq \tau \) and \( m_{\supp(\tau)} \leq m_{\sigma} - \sigma \). In this note we prove that the incidence matrix of all shape \( \sigma \) subspaces versus all shape \( \tau \) subspaces in the projective Hjelmslev geometry \( \text{PHG}(R^n) \) has full rank over the real numbers.

Keywords projective Hjelmslev geometries · finite chain rings · modules over finite chain rings

Mathematics Subject Classification (2000) 51C05, 51E05, 51E23

1 Preliminaries

It is known that the rank of an incidence matrix of all \( s \)-dimensional versus all \( t \)-dimensional subspaces in \( \text{PG}(n, q) \), \( 0 \leq s < t \leq n - s - 1 \), is equal to the number of the \( s \)-spaces in the geometry [3]. In this paper, we prove a similar result for the incidence matrices of shape \( \sigma \) versus shape \( \tau \) subspaces in the projective Hjelmslev space \( \text{PHG}(R^n) \). \( R \) is a chain ring, under a mild restriction on the shapes \( \sigma \) and \( \tau \).

This research is done within the COST Action IC-1004 "Random Network Coding and Designs over GF(q)". The first author has been supported by the Bulgarian NSF under Contract L01/0003. This research has been done as a part of the work under the project "Finite geometries, coding theory and cryptography" between the Research Foundation - Flanders (FWO) and the Bulgarian Academy of Sciences.

I. Landjev
New Bulgarian University, Montevideo str. 21, 1618 Sofia, Bulgaria
E-mail: i.landjev@abu.bg

Peter Vandendriesche
Ghent University, Department of Mathematics (WE01), Krijgslaan 281-S22, 9000 Ghent, Belgium
E-mail: pv@cage.ugent.be
We start with some definitions and results about chain rings and modules over finite chain rings. Let $R$ be a finite chain ring with $|R| = q^m$, $R/Rad R \cong \mathbb{F}_q$, where $q = p^b$ with $p$ prime. It is known that all ideals of $R$ have the form $R\theta_i = \theta R$ for some $\theta \in R \setminus Rad R$ and some $i \in \{1, \ldots, m-1\}$. Denote by $\Gamma = \{\gamma_0 = 0, \gamma_1 = 1, \gamma_2, \ldots, \gamma_{q-1}\}$ a set of elements of $R$ no two of which are congruent modulo $Rad R$. Every element $r \in R$ can be written uniquely as $r = \sum_{i=0}^{m-1} r_i \theta_i$, where $r_i \in \Gamma$. Throughout the paper these letters will keep the meaning fixed above. For the basic properties of chain rings we refer to [2, 4, 6].

Let $R^M$ be a finitely generated (left) module over the chain ring $R$. There exists a uniquely determined partition $\lambda = (\lambda_1, \ldots, \lambda_k) = \log_q |M|$ into parts $\lambda_i \leq m$ such that
\[
R^M \cong R/(Rad R)^{\lambda_1} \oplus \cdots \oplus R/(Rad R)^{\lambda_k}.
\] (1)

The partition $\lambda$ is called the shape and the conjugate partition $\lambda' = (\lambda_1', \lambda_2', \ldots)$ is called the conjugate shape of $R^M$. The integer $k$ is called the rank of $R^M$.

For a given positive integer $n$ and a non-increasing sequence of non-negative integers $\kappa = (\kappa_1, \ldots, \kappa_n)$ we denote by $\mathcal{G}(n, \kappa)$ the set of all (left) submodules of shape $\kappa$ of $R^n$. In this note we shall be confined to left submodules. This is not a restriction since every left submodule can be considered as a right submodule over the opposite chain ring. In what follows, we denote the sequence $(m, \ldots, m, 0, \ldots, 0)$ by $m^n$. For two sequences $\lambda = (\lambda_1, \ldots, \lambda_n)$ and $\mu = (\mu_1, \ldots, \mu_n)$, we write $\lambda \preceq \mu$ if $\lambda_i \leq \mu_i$ for all $i = 1, \ldots, n$. For two integers $k, n$ with $k \leq n$, and a prime power $q$, $\left[ \begin{array}{l} n \\ k \end{array} \right]_q$ denotes the Gaussian coefficient:
\[
\left[ \begin{array}{l} n \\ k \end{array} \right] = \frac{(q^n - 1) \cdots (q^{n-k+1} - 1)}{(q - 1) \cdots (q - 1)}.
\]

In what follows, we shall need formulas for the number of the submodules of given shape $\mu$ contained in a fixed module of shape $\lambda$ over a finite chain ring $R$. For the special case of $R = \mathbb{Z}_{q^n}$ they are known from [1]. The case of a general chain ring $R$ is settled in [5, Chapter 2].

**Theorem 1** Let $R^M$ be a module of shape $\lambda$. For every partition $\mu$ satisfying $\mu \preceq \lambda$ the module $R^M$ has exactly
\[
\left[ \begin{array}{l} \lambda \\ \mu \end{array} \right]_q := \prod_{i=1}^{n} q^{\mu_{i+1} - \mu_i} \left[ \begin{array}{l} \lambda_i - \mu_{i+1} \\ \mu_i - \mu_{i+1} \end{array} \right]_q
\]
submodules of shape $\mu$.

By duality, Theorem 1 allows to find the number of shape $\lambda$ submodules of $R^n$ contained in a fixed submodule of shape $\mu$. One has to apply Theorem 1 to the dual submodules that have shapes $m^n - \lambda$ and $m^n - \mu$, respectively.
The (left) projective Hjelmslev geometries $\text{PHG}(R^n)$ are produced from the finite modules $R^n$ in the same way one produces the classical projective geometries $\text{PG}(n-1, q)$ from the vector spaces $\mathbb{F}_q^n$. The geometry $\text{PHG}(R^n)$ is defined as an incidence structure $(\mathcal{P}, \mathcal{I}, I)$ having as points the free rank 1 submodules of $R^n$ and as lines the free rank 2 submodules of $R^n$. Incidence is given by set-theoretical inclusion. The set of points contained in a submodule $RM \subset R^n$ which is of shape $\lambda$ is called a subspace of shape $\lambda$. The subspaces defined by free submodules of $R^n$ are called Hjelmslev subspaces.

Two subspaces $L$ and $M$ defined by the modules $R\ell$ and $R\ell$, respectively, are called $i$-neighbors if $R\ell = R\ell + \theta R^n$. This is denoted by $L \sim_i M$. It can be checked that $i$-neighborhood is an equivalence relation on the set of all subspaces. The equivalence class of all subspaces that are $i$-neighbors to $L$ is denoted by $[L]^{(i)}$. Set

$$\mathcal{P}^{(i)} = \{ [x]^{(i)} | x \in \mathcal{P} \}, \mathcal{L}^{(i)} = \{ [L]^{(i)} | L \in \mathcal{L} \},$$

and

$$I^{(i)} = \{ ([x]^{(i)}, [L]^{(i)}) | \exists x' \in [x]^{(i)}, L' \in [L]^{(i)} : (x', L') \in I \}.$$

**Theorem 2** The incidence structure $(\mathcal{P}^{(i)}, \mathcal{L}^{(i)}, I^{(i)})$ is isomorphic to the geometry $\text{PHG}(R^n)$. 

Let us note that $R/\theta R$ is again a chain ring of size $q^n$, with residue field $\mathbb{F}_q$. In particular, for $i = 1$ we get $(\mathcal{P}^{(1)}, \mathcal{L}^{(1)}, I^{(1)}) \cong \text{PG}(n-1, q)$.

2 The Theorem

Let $R$ be a finite chain ring with $|R| = q^n$, $R/\text{Rad} R \cong \mathbb{F}_q$, and let $\Omega = \text{PHG}(R^n)$. Let further $\sigma = (\sigma_1, \ldots, \sigma_n)$ and $\tau = (\tau_1, \ldots, \tau_n)$ be non-increasing sequences of non-negative integers, i.e. $m \geq \sigma_1 \geq \ldots \geq \sigma_n \geq 0$, $m \geq \tau_1 \geq \ldots \geq \tau_n \geq 0$, with $\sigma \not\leq \tau$. We denote by $\text{supp}(\sigma)$ the set of indices $j$ for which $\sigma_j \neq 0$. We define a $\mathbb{F}_q$-matrix $M_{\sigma, \tau}$ in which the rows are indexed by the elements $G(n, \sigma)$ and columns are indexed by the elements of $G(n, \tau)$. The element $m(S, T)$ which is in the row indexed by $S \in G(n, \sigma)$ and the column indexed by $T \in G(n, \tau)$ is defined by

$$m(S, T) = \begin{cases} 1 & \text{if } S \subset T, \\ 0 & \text{if } S \not\subset T. \end{cases}$$

We denote by $\rho(S)$ the row of $M_{\sigma, \tau}(\Omega)$ indexed by the shape $\sigma$ subspace $S$. Our goal is to prove the following theorem which is an analog of Kantor's result [3] about the rank of the incidence matrix of dimension $s$ versus dimension $t$ subspaces in $\text{PG}(n-1, q)$.

**Theorem 3** Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ and $\tau = (\tau_1, \ldots, \tau_n)$ be two non-increasing sequences of non-negative integers with $\sigma \not\leq \tau$ and $m^{(\text{supp}(\tau))} \leq m^n - \sigma$. Then the rank of $M_{\sigma, \tau}(\Omega)$ is equal to the number of shape $\sigma$ subspaces of $\Omega$, i.e. $[m^n]_{\sigma, \tau}$.
Let us note that while the theorem covers the most important cases when \( \sigma = m^s \), or \( \tau = m^t \), or both, it does not cover the case where there is no \( k \) with \( \sigma \leq m^k \leq \tau \). This case will require an additional argument.

3 The Proof of Theorem 3

The proof will be made in several steps. First, we consider the case when the modules that index the rows and the columns are free, i.e. \( \sigma = m^s \) and \( \tau = m^t \).

**Lemma 1** Let \( R \) be a chain ring with \( |R| = q^m \), \( R/\text{Rad} R \cong \mathbb{F}_q \), and let \( \Omega = \text{PHG}(R R^m) \). Let further \( s \) and \( t \) be integers with \( 1 \leq s \leq t \leq n - s \) and \( \sigma = m^s \), \( \tau = m^t \). Then the rank of \( M_{\sigma,\tau}(\Omega) \) is equal to the number of free Hjelmslev subspaces of \( \Omega \) of dimension \( s - 1 \) i.e. the rank is equal to \( [m^s]_q \).

**Proof** We use induction on \( m \). The case \( m = 1 \) is Kantor’s Theorem. Now let us assume that the result is proved for all incidence matrices \( M_{\sigma,\tau}(\Omega') \) where \( \Omega' \) is an \( (n - 1) \)-dimensional projective Hjelmslev geometry over a chain ring of nilpotency index at most \( m - 1 \).

Now let \( R \) be a chain ring with \( |R| = q^m \), \( q = p^k \), \( R/\text{Rad} R \cong \mathbb{F}_q \), and denote \( \Omega = \text{PHG}(R R^m) \). Consider two \((m - 1)\)-neighbor classes of Hjelmslev subspaces of shape \( \sigma = m^s \) and \( \tau = m^t \), say \( [S]^{(m-1)} = \{S_1, \ldots, S_u\} \) and \( [T]^{(m-1)} = \{T_1, \ldots, T_v\} \), respectively. If some subspace from \([S]\) contains a point which is not incident with a subspace from \([T]^{(m-1)}\) then \( S_i \not\subset T_j \) for any \( i \in \{1, \ldots, u\} \) and any \( j \in \{1, \ldots, v\} \). Hence the \( u \times v \) submatrix of \( M_{\sigma,\tau}(\Omega) \) defined by the rows indexed by \( S_1, \ldots, S_u \) and the columns indexed by \( T_1, \ldots, T_v \) is the all-zero matrix. Otherwise, each subspace of \([S]^{(m-1)}\) is contained in the same number of subspaces from \([T]^{(m-1)}\) and each subspace from \([T]^{(m-1)}\) contains the same number of subspaces from \([S]^{(m-1)}\). Hence the submatrix of \( M_{\sigma,\tau}(\Omega) \) with rows indexed by the subspaces from \([S]^{(m-1)}\) and the columns indexed by the subspaces from \([T]^{(m-1)}\) is a \((0,1)\)-matrix, \( B \) say, with constant row and column sums. For a suitable ordering of all Hjelmslev subspaces of dimension \( s - 1 \), resp. \( t - 1 \), the matrix \( M_{\sigma,\tau}(\Omega) \) can be represented in the following block form:

\[
M_{\sigma,\tau}(\Omega) = (A_{i,j})_{x \times y},
\]

Here \( x \) and \( y \) are the numbers of neighbor classes of subspaces of dimension \( s - 1 \), resp. \( t - 1 \). By Theorem 2, we get

\[
x = q^{(m-1)(m-s)} \binom{n}{s}_q, \quad y = q^{(m-1)(m-t)} \binom{n}{t}_q.
\]

If the \( i \)-th \((m - 1)\)-neighbor class of \( s - 1 \) dimensional Hjelmslev subspaces is contained in the \( j \)-th \((m - 1)\)-neighbor class of \( t - 1 \)-dimensional Hjelmslev subspaces in the factor geometry then \( A_{i,j} \) is a \( u \times v \) \((0,1)\)-matrix which has the form \( PBQ \) for some suitable permutation matrices \( P \) and \( Q \) of orders \( u \) and \( v \).
and \( v \), respectively. Otherwise \( A_{i,j} \) is the zero matrix. Moreover, the \( x \times y \) matrix \( A = (a_{i,j})_{x \times y} \) defined by

\[
a_{i,j} = \begin{cases} 
1 & \text{if } A_{i,j} \neq 0_{x \times y}, \\
0 & \text{if } A_{i,j} = 0_{x \times y},
\end{cases}
\]

is equivalent to the incidence matrix of free \((s-1)\)-dimensional vs. free \((t-1)\)-dimensional Hjelmslev subspaces in the factor geometry \( PHG(R/R\theta) \). Since \( R/R\theta \) has nilpotency index \( m-1 \), the rank of \( A \) is equal to the number of its rows, by the induction hypothesis.

Assume there exists a non-trivial linear combination of the rows of \( M_{\sigma,t}(\Omega) \).

\[
\sum_{S} a(S)\rho(S) = \sum_{[S]^{(m-1)}} \sum_{L \in [S]^{(m-1)}} a(L)\rho(L) = 0, \tag{3}
\]

where \( a(L) \) are real numbers not all zero. Define

\[
G = \{ I + C\theta^{m-1} \mid C \text{ is an } n \times n \text{ matrix over } \Gamma \text{ with } 0\text{'s on the main diagonal} \}.
\]

\( G \) is a commutative group under matrix multiplication. \( G \) fixes all \((m-1)\)-neighbor classes of points setwise and acts transitively on the points within these classes. Hence the orbits of \( G \) on the set of all Hjelmslev subspaces are the \((m-1)\)-neighbor classes of Hjelmslev spaces themselves. In particular, this is true for all \((s-1)\)-dimensional Hjelmslev subspaces. Thus for every \((s-1)\)-dimensional Hjelmslev subspace \( S \) we have

\[
|G_{S}| \cdot |S^{G}| = |G|.
\]

Since all orbits \( S^{G} \) have the same size, the stabilizers \( G_{S} \) have also the same size. For an arbitrary \( g \in G \), we get from (3):

\[
\sum_{[S]^{(m-1)}} \sum_{L \in [S]^{(m-1)}} a(L)\rho(L^{g}) = 0.
\]

Let \( g \) run over all elements of \( G \). This implies

\[
\sum_{g \in G} \sum_{[S]^{(m-1)}} \sum_{L \in [S]^{(m-1)}} a(L)\rho(L^{g}) = \sum_{[S]^{(m-1)}} \sum_{L \in S} \sum_{g \in G} a(L)\rho(L) = 0.
\]

If \( |L| = |M| \) the number of elements \( g \in G \) for which \( L^{g} = M \) is equal to the size of the stabilizer of \( L \), i.e. \( |G_{L}| = |G|/|L^{G}| \), and is hence constant for all Hjelmslev subspaces of the same dimension. Hence there exist coefficients \( b([S]^{(m-1)}) \) such that

\[
\sum_{[S]^{(m-1)}} b([S]^{(m-1)}) \left( \sum_{L \in [S]^{(m-1)}} \rho(L) \right) = 0.
\]

Let the rows of the incidence matrix of \((s-1)\)-dimensional vs. \((t-1)\)-dimensional subspaces of \( PHG(R/R\theta) \) be \( r_{1}, \ldots, r_{y} \). For a suitable ordering of the
\[(s - 1)\text{-dimensional } \text{Hjelmslev subspaces of } \Omega \text{ and of the } (s - 1)\text{-dimensional subspaces of } \text{PG}(R/R\Theta) \text{ we get}
\[
\sum_{L \in [S_i]^{(m - 1)}} \rho(L) = k(r_i \otimes (1, \ldots, 1)).
\]

Here \(k\) denotes the number of ones in any column of the block \(B\) defined above. This implies that
\[
\sum_{[S]^{(m - 1)}} b([S]^{(m - 1)}) \sum_{L \in [S]^{(m - 1)}} \rho(L) = \sum_{i=1}^{x} b_i \cdot k(r_i \otimes (1, \ldots, 1)) = 0,
\]

where \(b_i = b([S_i]^{(m - 1)}) \cdot k\). Hence
\[
\sum_{i=0}^{x} b_i r_i = 0,
\]

a contradiction since by the induction hypothesis the rows \(r_i\) are linearly independent.

**Corollary 1** Let \(\sigma = m^s\) and let \(\tau\) be an arbitrary sequence with \(\sigma \leq \tau \leq m^n - \sigma\). Then \(M_{\sigma, \tau}\) is of full rank.

**Proof** Let \(t\) be the rank of the smallest free submodule of \(R^n\) that contains a submodule of shape \(\tau\). By \(\sigma \leq \tau \leq m^n - \sigma\), we get \(s \leq t \leq n - s\). Now we have
\[
M_{m^s, m^t} = \alpha M_{m^s, \tau} M_{\tau, m^t},
\]
where \(\alpha\) is the number of submodules \(U\) of shape \(\tau\) with \(S \subseteq U \subseteq T\), where \(S\) and \(T\) are fixed free submodules of ranks \(s\) and \(t\), respectively (hence \(\alpha\) is a constant). Since \(M_{m^s, m^t}\) is of full rank (by Lemma 1) then \(M_{m^s, \tau}\) is also of full rank by Sylvester’s inequality.

In a similar way one can prove the following lemma.

**Lemma 2** Let \(\tau = m^t\) and let \(\sigma\) be an arbitrary sequence with \(\sigma \leq \tau \leq m^n - \sigma\). Then \(M_{\sigma, \tau}\) is of full rank.

**Idea of proof.** Using the idea from the proof of Corollary 1, we can assume that \(t = |\text{supp}(\sigma)|\). Then the proof proceeds by induction on \(m\), the case \(m = 1\) being again Kantor’s theorem.

Next we assume that the result has been proved for geometries over chain rings of nilpotency index at most \(m - 1\) and consider \(\Omega = \text{PG}(R/R^m)\), where \(R\) is a chain ring of nilpotency index \(m\). Let \(S\) be a subspace of shape \(\sigma\) in \(\Omega\) and \(T_1, T_2\) - Hjelmslev subspaces of shape \(\tau\) in \(\Omega\) with \(S \subseteq T_1, S \subseteq T_2\). Clearly \(T_1\) and \(T_2\) are Hjelmslev subspaces of minimal rank containing \(S\). This implies that \(T_1\) and \(T_2\) are neighbours; otherwise the Hjelmslev subspace \(T_1 \cap T_2\) would contain \(S\) which is a contradiction to the minimality of \(T_1\) and \(T_2\). This
implies that there exists such ordering of the shape $\sigma$ and shape $\tau$ subspaces that $M_{\sigma,\tau}$ has diagonal block form with zero-blocks off the main diagonal. The diagonal blocks have full rank by Theorem 2 and by the induction hypothesis.

Now Theorem 3 follows from Lemma 1 and Lemma 2.

Acknowledgments. The authors thank the referees for the critical reading of the manuscript and the numerous remarks and suggestions.

References

On weighted $\{\delta v_{\mu+1}, \delta v_{\mu}; k-1, q\}$-minihypers, $q$ square

L. Beukemann · K. Metsch · L. Storme

Received: date / Accepted: date

Abstract Weighted minihypers have recently received a lot of attention. They originated as geometrical equivalents of linear codes meeting the Griesmer bound, but have also been investigated for their importance in solving geometrical problems. Storme characterized weighted $\{\delta(q+1), \delta; k-1, q\}$-minihypers, $q$ square, as a sum of lines and Baer subgeometries $\text{PG}(3, \sqrt{q})$, provided $\delta$ is sufficiently small. This result is the basis for a new characterization result on weighted $\{\delta v_{\mu+1}, \delta v_{\mu}; k-1, q\}$-minihypers. We show that such minihypers are a sum of $\mu$-dimensional subspaces and of (projected) $(2\mu+1)$-dimensional Baer subgeometries $\text{PG}(2\mu+1, \sqrt{q})$. Because of the equivalence of minihypers with linear codes meeting the Griesmer bound, a new contribution to the characterization of linear codes meeting the Griesmer bound is also obtained.

Keywords Griesmer bound · Minihypers · Weighted sets · Baer subgeometries

Mathematics Subject Classification (2000) 05B25 · 51E20 · 51E21 · 94B05

1 Introduction

1.1 The Griesmer bound

A linear $[n, k, d]$-code $C$ over the finite field $\mathbb{F}_q$ of order $q$ is a $k$-dimensional subspace of the $n$-dimensional vector space $V(n, q)$ of vectors of length $n$ over $\mathbb{F}_q$. The minimum distance $d$ of the code $C$ is the minimal number of positions in which two distinct codewords of $C$ differ [12].

L. Beukemann
Justus-Liebig-Universität, Mathematisches Institut, Arndtstraße 2, D-35392 Giessen, Germany
E-mail: linda.beukemann@math.uni-giessen.de

K. Metsch
Justus-Liebig-Universität, Mathematisches Institut, Arndtstraße 2, D-35392 Giessen, Germany
E-mail: klaus.metsch@math.uni-giessen.de

L. Storme
Ghent University, Dept. of Mathematics, Krijgslaan 281-S22, 9000 Ghent, Belgium
E-mail: ls@cage.ugent.be
Between the parameters $n, k, d$ of a linear $[n, k, d]$-code $C$, many relations and bounds exist. One of these bounds is the Griesmer bound.

The Griesmer bound states that if there exists an $[n, k, d]$-code over $\mathbb{F}_q$ for given values of $k, d,$ and $q$, then

$$n \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil = g_q(k, d),$$

where $\left\lceil x \right\rceil$ denotes the smallest integer larger than or equal to $x$ [6,16].

Considering this lower bound on the length $n$ for given values $k, d,$ and $q$, the question arises whether there exists a linear $[n, k, d]$-code whose length $n$ is equal to the lower bound $g_q(k, d)$.

We contribute to the theory of linear codes meeting the Griesmer bound, by using the geometrical equivalent objects, called minihypers.

1.2 Definition

Let $\text{PG}(N, q)$ be the $N$-dimensional projective space over the finite field of order $q$. Let $v_{N+1} = (q^{N+1} - 1)/(q - 1)$ denote the number of points of $\text{PG}(N, q)$.

**Definition 1** (Hamada, Helleseth, and Tamari [8,10]) An $\{f, m; N, q\}$-minihyper is a pair $(F, w)$, where $F$ is a subset of the point set of $\text{PG}(N, q)$ and where $w$ is a weight function $w : \text{PG}(N, q) \rightarrow \mathbb{N} : x \mapsto w(x)$, satisfying

1. $w(x) > 0 \iff x \in F$,
2. $\sum_{x \in F} w(x) = f$, and
3. $\min(\sum_{x \in H} w(x) | H \in \mathcal{H}) = m$; where $\mathcal{H}$ denotes the set of hyperplanes of $\text{PG}(N, q)$.

In the case that $w$ is a mapping onto $\{0, 1\}$, the minihyper $(F, w)$ can be identified with the set $F$ and is simply denoted by $F$.

This definition shows that an $\{f, m; N, q\}$-minihyper is a particular $m$-fold blocking set with respect to the hyperplanes of $\text{PG}(N, q)$, a topic investigated in great detail in Galois geometries [11]. This link with (multiple) blocking sets has already proven to be very useful in obtaining new characterization results on minihypers.

1.3 Link between minihypers and linear codes meeting the Griesmer bound

Suppose that there exists a linear $[n, k, d]$-code over $\mathbb{F}_q$ meeting the Griesmer bound $(d \geq 1, k \geq 3)$, then we can write $d$ in an unique way as $d = \theta q^{k-1} - \sum_{i=0}^{k-2} \epsilon_i q^i$ such that $\theta \geq 1$ and $0 \leq \epsilon_i < q$.

Using this expression for $d$, the Griesmer bound for an $[n, k, d]$-code can be expressed as $n \geq \theta v_k - \sum_{i=0}^{k-2} \epsilon_i v_{i+1}$.

It follows from Hamada and Tamari [10], see also Hamada and Helleseth [8], that there is a one-to-one correspondence between the set of all non-equivalent $[n, k, d]$-codes meeting the Griesmer bound and the set of all projectively distinct $\{\sum_{i=0}^{k-2} \epsilon_i v_{i+1}, \sum_{i=0}^{k-2} \epsilon_i v_i; k-1, q\}$-minihypers $(F, w)$, such that $1 \leq w(P) \leq \theta$ for every point $P \in F$. 
More precisely, the link is described in the following way.

Let \( G = (g_1, \ldots, g_n) \) be a generator matrix for a linear \([n, k, d]\)-code, meeting the Griesmer bound. We look at a column of \( G \) as being the coordinates of a point in \( \text{PG}(k-1, q) \). Let the point set of \( \text{PG}(k-1, q) \) be \( \{s_1, \ldots, s_{v_k}\} \). Let \( m_i(G) \) denote the number of columns in \( G \) defining \( s_i \). Let \( m(G) \) be the maximum value in \( \{m_i(G) \mid i = 1, 2, \ldots, v_k\} \). Then \( \theta = m(G) \) is uniquely determined by the code \( C \) and we call it the maximum multiplicity of the code. Define the weight function \( w : \text{PG}(k-1, q) \to \mathbb{N} \) by \( w(s_i) = \theta - m_i(G) \), \( i = 1, 2, \ldots, v_k \). Let \( F = \{s_i \in \text{PG}(k-1, q) \mid w(s_i) > 0\} \), then \( (F, w) \) is a \( \{\sum_{i=0}^{k-2} \epsilon_i v_{i+1}, \sum_{i=0}^{k-2} \epsilon_i v_i; k-1, q\} \)-minihyper.

1.4 The Belov, Logachev, and Sandimirov construction

Now the question arises how to construct linear codes meeting the Griesmer bound. The standard construction method is of Belov, Logachev, and Sandimirov [1]. This construction method is easily described by using the corresponding minihypers. Consider in \( \text{PG}(k-1, q) \) a sum of

- \((0)\) \( \epsilon_0 \) points \( P_1, \ldots, P_{\epsilon_0} \),
- \((1)\) \( \epsilon_1 \) lines \( \ell_1, \ldots, \ell_{\epsilon_1} \),
- \((k-2)\) \( \epsilon_{k-2} \) \((k-2)\)-dimensional subspaces \( \pi_1^{(k-2)}, \ldots, \pi_{\epsilon_{k-2}}^{(k-2)} \),

with \( 0 \leq \epsilon_i \leq q-1, i = 0, \ldots, k-2 \).

Then such a set defines a \( \{\sum_{i=0}^{k-2} \epsilon_i v_{i+1}, \sum_{i=0}^{k-2} \epsilon_i v_i; k-1, q\} \)-minihyper, where the weight \( w(P) \) of a point \( P \) is equal to the number of objects, in the description above, with which it is incident.

1.5 Results of Hamada, Helleseth, and Maekawa

Now that the standard examples of minihypers, so equivalently, of linear codes meeting the Griesmer bound, are known, the characterization problem on minihypers, so equivalently, on linear codes meeting the Griesmer bound, arises.

Characterize \( \{f, m; k-1, q\} \)-minihypers for given parameters \( f = \sum_{i=0}^{k-2} \epsilon_i v_{i+1} \), \( m = \sum_{i=0}^{k-2} \epsilon_i v_i, k \), and \( q \).

Fundamental research on this characterization problem was performed by Hamada and Helleseth who, in a large number of articles, developed an extensive theory on minihypers and who developed a large amount of techniques useful in the study of minihypers.

Their main results are:
Theorem 1 (Hamada, Helleseth, and Maekawa [7, 9]) A \( \{ \sum_{i=0}^{k-2} \epsilon_i v_{i+1}; \sum_{i=0}^{k-2} \epsilon_i v_i; k-1, q \} \)-minihyper, where \( \sum_{i=0}^{k-2} \epsilon_i < \sqrt{q} + 1 \), is a union of \( \epsilon_{k-2} \) hyperplanes, \( \epsilon_{k-3} \) \((k - 3)\)-dimensional spaces, \( \ldots \), \( \epsilon_1 \) lines, and \( \epsilon_0 \) points, which all are pairwise disjoint, so is of Belov-Logachev-Sandimirov type.

A particular type of minihypers, of interest for the solution of many geometrical problems, is the class of \( \{ \delta v_{\mu+1}, \delta v_{\mu}; N, q \} \)-minihypers. Here, the most important characterization result on non-weighted \( \{ \delta v_{\mu+1}, \delta v_{\mu}; N, q \} \)-minihypers was obtained by Govaerts and Storme [4].

Theorem 2 (Govaerts and Storme [4]) A \( \{ \delta v_{\mu+1}, \delta v_{\mu}; N, q \} \)-minihyper \( F, q > 16 \) square, \( \delta < q^{5/8}/\sqrt{2} + 1 \), \( 2\mu + 1 \leq N \), is a unique union of \( \mu \)-spaces and subgeometries \( PG(2\mu + 1, \sqrt{q}) \), which are pairwise disjoint.

2 Weighted minihypers

Recently, there has been a detailed study of weighted minihypers \( (F, w) \). This allows for the characterization of minihypers involving more geometrical objects.

The problem of characterizing weighted minihypers \( (F, w) \) not only involves the determination of the set of points \( F \) belonging to the minihyper, it also involves the determination of the precise weights of the points of the minihyper \( (F, w) \).

De Beule, Metsch, and Storme obtained in [3] the weighted version of the results of Hamada, Helleseth, and Maekawa of Theorem 1.

Theorem 3 (De Beule, Metsch, and Storme [3]) A weighted \( \{ \sum_{i=0}^{k-2} \epsilon_i v_{i+1}, \sum_{i=0}^{k-2} \epsilon_i v_i; k-1, q \} \)-minihyper \( (F, w) \), with \( \sum_{i=0}^{k-2} \epsilon_i < \sqrt{q} + 1 \), is a sum of \( \epsilon_{k-2} \) hyperplanes, \( \epsilon_{k-3} \) \((k - 3)\)-dimensional spaces, \( \ldots \), \( \epsilon_1 \) lines, and \( \epsilon_0 \) points, so is of Belov-Logachev-Sandimirov type.

But similarly, research has focussed on determining a weighted version of the characterization result for \( \{ \delta v_{\mu+1}, \delta v_{\mu}; N, q \} \)-minihypers of Theorem 2. Namely, Storme proved the following result.

Theorem 4 (Storme [17]) Let \( w \) be a weighted \( \{ \delta(q + 1), \delta; k-1, q \} \)-minihyper, with \( q = p^{2h} \), \( p \) prime, \( p \geq 11 \), \( k \geq 3 \), \( \delta \leq (q - 1)/4 \) and if \( h > 1 \) also with

\[
\delta + \frac{\delta^2}{q} + \frac{2\delta^2 - \delta}{q^2} + \frac{\delta^2 - \delta}{q^3} < 1 + \epsilon,
\]

where \( q + 1 + \epsilon \) is the size of the smallest minimal blocking sets in \( PG(2, q) \) different from a line and a Baer subplane. Then \( w \) is a sum of lines and of Baer subgeometries \( PG(3, \sqrt{q}) \).

In the preceding theorem, bounds on small minimal blocking sets in \( PG(2, q) \) are used.

Definition 2 A blocking set \( B \) of \( PG(2, q) \) is a set of points intersecting every line in at least one point.

A blocking set \( B \) of \( PG(2, q) \) is called minimal when no proper subset of \( B \) still is a blocking set.

A blocking set \( B \) of \( PG(2, q) \) is called small when \(|B| < 3(q+1)/2|\).
On weighted $\{\delta v_{\mu+1}, \delta v_\mu; k-1, q\}$-minihypers, $q$ square

Small minimal blocking sets have been investigated in great detail. The results of Sziklai and Szönyi prove that every line of $\text{PG}(2, q)$ intersects a small minimal blocking set $B$ of $\text{PG}(2, q)$, $q = p^h$, $p$ prime, $h \geq 1$, in $1 \mod p$ points. Moreover, if $e$ is the largest integer such that every line intersects $B$ in $1 \mod p^e$ points, then $e$ divides $h$, and every $(1+p^e)$-secant to $B$ intersects $B$ in a subline $\text{PG}(1, p^e)$ [18,19]. This largest integer $e$ for which every line of $\text{PG}(2, q)$ intersects a small minimal blocking set $B$ in $1 \mod p^e$ points is called the exponent of $B$.

All the small minimal blocking sets of $\text{PG}(2, q)$, $q = p^h$, $p$ prime, $h \geq 1$, with exponent $e \geq h/3$, have been characterized. They are either a line ($e = h$), a Baer subplane $\text{PG}(2, \sqrt{q})$ ($e = h/2$ for $h$ even) [19], or a projected subgeometry $\text{PG}(3, q^{1/3})$ in $\text{PG}(2, q)$ ($e = h/3$ when $h$ is a multiple of $3$) [13–15].

Moreover, a small minimal blocking set $B$ of $\text{PG}(2, q)$, $q = p^h$, $p$ prime, $h \geq 1$, with exponent $e$, has size $|B| \geq q + 1 + \frac{4}{h+2}$.

These preceding results on the small minimal blocking sets of $\text{PG}(2, q)$, $q = p^h$, $p$ prime, $h \geq 1$, are the crucial tools in obtaining new results on minihypers.

Theorem 4 was the basis for the following new characterization result on weighted minihypers.

**Theorem 5** (Beukemann, Metsch, and Storme [2]) Let $w$ be a weighted $\{\delta v_{\mu+1}, \delta v_\mu; k-1, q\}$-minihyper, with $q = p^{2h}$, $p$ prime, $p \geq 11$, $k \geq 4$, $1 \leq \mu \leq k-2$, $\delta \leq (q-1)/4$ and if $h > 1$ also with

\[
\delta + \frac{\delta^2}{q^2} + \frac{2\delta^2 - \delta}{q^4} + \frac{\delta^2 - \delta}{q^6} < 1 + \epsilon,
\]

where $q + 1 + \epsilon$ is the size of the smallest minimal blocking sets in $\text{PG}(2, q)$ different from a line and a Baer subplane. Then $w$ is a sum of $\mu$-spaces $\text{PG}(\mu, q)$ and of (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$.

Before presenting the main ideas which led to this characterization result, we first discuss what a projected Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ looks like to give a precise idea of what the characterized minihypers look like.

A projected Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ is a cone with an $s$-dimensional vertex, $s \geq 0$, and base a naturally embedded Baer subgeometry $\text{PG}(2\mu - 2s - 1, \sqrt{q})$. Here, the points of the $s$-dimensional vertex have weight $\sqrt{q} + 1$, while all the other points of the cone have weight one.

2.1 The inductive characterization methods

Theorem 5 is first of all proven by induction on $\mu$. But secondly, to prove that the weighted minihyper $(F, w)$ is a sum of $\mu$-spaces $\text{PG}(\mu, q)$ and of (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$, these different components must be determined one by one. Here, the known result is applied that if a subspace $\Pi = \text{PG}(\mu, q)$ is contained in a weighted $\{\delta v_{\mu+1}, \delta v_\mu; k-1, q\}$-minihyper $(F, w)$, $\delta \leq (q-1)/2$, then $(F, w) - \Pi$ is a weighted $\{(\delta - 1)v_{\mu+1}, (\delta - 1)v_\mu; k-1, q\}$-minihyper. The similar result that if a (projected) Baer subgeometry $\Omega = \text{PG}(2\mu + 1, \sqrt{q})$ is contained in a weighted $\{\delta v_{\mu+1}, \delta v_\mu; k-1, q\}$-minihyper $(F, w)$, $\delta \leq (q-1)/2$, then $(F, w) - \Omega$ is a weighted $\{(\delta - (\sqrt{q}+1))v_{\mu+1}, (\delta - (\sqrt{q}+1))v_\mu; k-1, q\}$-minihyper is proven in [2].
Additionally, a sum of $\mu$-dimensional spaces $\text{PG}(\mu, q)$ and of (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$ defining a $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$, with $\delta \leq (q - 1)/4$, may have coinciding $\mu$-dimensional spaces or coinciding (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$. For instance, $\alpha' > 0$ times the same $\mu$-dimensional space in a $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$ contributes $\alpha'$ to the parameter $\delta$ of this minihyper $(F, w)$, and $\alpha'' > 0$ times the same (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ in a $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$ contributes $\alpha''(\sqrt{q} + 1)$ to the parameter $\delta$ of this minihyper $(F, w)$. To be able to find the $\mu$-dimensional spaces and the (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$ contained in the $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$ with the correct multiplicity, we work with the minimal weight $\alpha$ of the points of the $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$.

A point with minimal weight $\alpha$ of the $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$ either lies in a $\mu$-dimensional space $\text{PG}(\mu, q)$ contained in $(F, w)$ with multiplicity $\alpha$, or in a (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ contained in $(F, w)$ with multiplicity $\alpha$.

By first reconstructing these $\mu$-dimensional spaces and (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$, with the smallest multiplicity $\alpha$, contained in $(F, w)$, it is possible to reconstruct these spaces and (projected) Baer subgeometries with their correct multiplicity $\alpha$. They are then removed $\alpha$ times to remain with either a $(\{\delta - \alpha\} v_{\mu + 1}, (\delta - \alpha)v_{\mu}; k - 1, q)$-minihyper in the case of a $\mu$-dimensional space or with a $(\{(\delta - \alpha)(\sqrt{q} + 1)\} v_{\mu + 1}, (\delta - \alpha(\sqrt{q} + 1))v_{\mu}; k - 1, q)$-minihyper in the case of a (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$. Then a new $\mu$-dimensional space or (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ with the smallest multiplicity in the remaining minihyper is reconstructed.

2.2 Reconstructing the (projected) Baer subgeometries with the smallest multiplicity

It is required to use different techniques to reconstruct the $\mu$-spaces and the (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$, with the smallest multiplicity $\alpha$, contained in the weighted $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihypers $(F, w)$.

The (projected) Baer subgeometries $\text{PG}(2\mu + 1, \sqrt{q})$ contained in the weighted $\{\delta v_{\mu + 1}, \delta v_{\mu}; k - 1, q\}$-minihyper $(F, w)$ are reconstructed before the $\mu$-dimensional spaces contained in $(F, w)$ are reconstructed. These (projected) Baer subgeometries contained in $(F, w)$ are determined in the following way.

First of all, as mentioned above, induction on $\mu$ is used. Consider a $(k - 3)$-dimensional space intersecting $(F, w)$ in a $\{\delta v_{\mu - 1}, \delta v_{\mu}; k - 3, q\}$-minihyper. Then all the hyperplanes $H$ through this $(k - 3)$-dimensional space intersect $(F, w)$ in a weighted $\{\delta v_{\mu - 1}, \delta v_{\mu}; k - 2, q\}$-minihyper $(F_H, w_H)$. By using the induction hypothesis on $\mu$, such a hyperplane $H$ intersects $(F, w)$ in a sum of $(\mu - 1)$-dimensional spaces and of (projected) Baer subgeometries $\text{PG}(2\mu - 1, \sqrt{q})$.

If there is a (projected) Baer subgeometry $\text{PG}(2\mu - 1, \sqrt{q})$ contained in $(F_H, w_H)$, then this means that there is a (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ contained in the minihyper $(F, w)$. This latter (projected) Baer subgeometry $\text{PG}(2\mu + 1, \sqrt{q})$ contained in the minihyper $(F, w)$ is reconstructed in the following way.

We isolate a particular (projected) Baer subgeometry $\text{PG}(2\mu - 2, \sqrt{q})$ of $(F_H, w_H)$ of minimal weight $\alpha$ in a $(k - 3)$-dimensional subspace $\Delta$ of $H$. The particular se-
lection of $\Delta$ assures that many hyperplanes $H_i$ through $\Delta$ intersect $(F, w)$ in a \{\delta v_{\mu}, \delta v_{\mu-1}; k - 2, q\}-minihyper. Because of the induction hypothesis on $\mu$, the intersections of these hyperplanes $H_i$ with $(F, w)$ are characterized, and this leads to (projected) Baer subgeometries PG(2$\mu - 1, \sqrt{q}$)$_i$ contained in $(F, w) \cap H_i$, and having minimal weight \( \alpha \), through the carefully selected (projected) Baer subgeometry PG(2$\mu - 2, \sqrt{q}$)$_i$ of $(F_H, w_H)$ in $\Delta$.

But since two of those (projected) Baer subgeometries PG(2$\mu - 1, \sqrt{q}$)$_i$, contained in $(F, w) \cap H_i, i = 0, \ldots, q$, let us denote them by PG(2$\mu - 1, \sqrt{q}$)$_1$ and PG(2$\mu - 1, \sqrt{q}$)$_2$, share this particularly isolated (projected) Baer subgeometry PG(2$\mu - 2, \sqrt{q}$)$_i$ in the $(k - 3)$-dimensional subspace $\Delta$ of $H$, they together define a (projected) Baer subgeometry PG(2$\mu, \sqrt{q}$)$_i$. It is proven that a large subset of this (projected) Baer subgeometry PG(2$\mu, \sqrt{q}$)$_i$ is contained in the minihyper $(F, w)$.

These (projected) Baer subgeometries PG(2$\mu, \sqrt{q}$)$_i$ then lead, via geometrical arguments, to a (projected) Baer subgeometry PG(2$\mu + 1, \sqrt{q}$), of minimal weight $\alpha$, contained in $(F, w)$.

2.3 Reconstructing the $\mu$-dimensional spaces

The $\mu$-dimensional spaces contained in a \{\delta v_{\mu+1}, \delta v_{\mu}; k - 1, q\}-minihyper $(F, w)$ are only reconstructed after all the (projected) Baer subgeometries PG(2$\mu + 1, \sqrt{q}$)$_i$ contained in $(F, w)$ have been determined.

This happens when we find a $(k - 3)$-dimensional space $\Delta'$ intersecting $(F, w)$ in a \{\delta v_{\mu-1}, \delta v_{\mu-2}; k - 3, q\}-minihyper, for which all the hyperplanes $H_i$ through $\Delta'$ intersect $(F, w)$ in a \{\delta v_{\mu}, \delta v_{\mu-1}; k - 2, q\}-minihyper which is a sum of $\delta$ $(\mu - 1)$-dimensional spaces.

Then the arguments of [5] can be used to prove that $(F, w)$ is a sum of $\delta$ $\mu$-spaces.

We briefly repeat these arguments to describe where different arguments had to be used to reconstruct in Subsection 2.2 the (projected) Baer subgeometries PG(2$\mu + 1, \sqrt{q}$)$_i$, of minimal weight $\alpha$, contained in $(F, w)$.

Again, as before, induction on $\mu$ is used. Consider a $(k - 3)$-dimensional space $\Delta'$ intersecting $(F, w)$ in a \{\delta v_{\mu-1}, \delta v_{\mu-2}; k - 3, q\}-minihyper, containing a point of minimal weight $\alpha$ of $(F, w)$. Then this point belongs to one $(\mu - 2)$-dimensional space $\Pi_{\mu - 2}$ of $\Delta'$, contained in $(F, w) \cap \Delta'$ and having weight $\alpha$ in $(F, w) \cap \Delta'$. Then all the hyperplanes $H_i, i = 0, \ldots, q$, through this $(k - 3)$-dimensional space $\Delta'$ intersect $(F, w)$ in a weighted \{\delta v_{\mu}, \delta v_{\mu-1}; k - 2, q\}-minihyper $(F_H, w_H)$. By using the induction on $\mu$ and by the assumption made in this subsection, such a hyperplane $H_i$ intersects $(F, w)$ in a sum of $(\mu - 1)$-dimensional spaces.

Because $\alpha$ is the minimal weight of the points of $(F, w)$, the intersections $H_i \cap (F, w)$ contain a $(\mu - 1)$-dimensional space $\Pi_{\mu - 1,i}$, of weight $\alpha$, passing through the space $\Pi_{\mu - 2}$. Consider two such $(\mu - 1)$-dimensional spaces $\Pi_{\mu - 1,i}$ and $\Pi_{\mu - 1,j}$. Since they share a $(\mu - 2)$-dimensional space, they define a $\mu$-dimensional space $\Pi_{\mu}$. It is proven that this $\mu$-dimensional space $\Pi_{\mu}$ is contained in the minihyper $(F, w)$, with weight $\alpha$.

In the preceding paragraph, we see that the difference between the dimension of the space $\Pi_{\mu - 2}$ and the spaces $\Pi_{\mu - 1,i}$ is equal to one. This makes it possible
to put these spaces \( \Pi_{\mu-1,i} \), together into the space \( \Pi_\mu \) of dimension \( \mu \), again one dimension larger.

In the preceding Subsection 2.2, for the case of the (projected) Baer subgeometries, if the same approach would be used, then the \((k-3)\)-dimensional space \( \Delta' \) intersecting \((F,w)\) in a \({\delta v_\mu-1, \delta v_\mu-2; k-3, q}\)-minihyper would share a (projected) Baer subgeometry \( PG(2\mu - 3, \sqrt{q}) \) with \((F,w)\), the hyperplanes \( H_i \) through \( \Delta' \) would intersect \((F,w)\) in a \({\delta v_\mu, \delta v_\mu-1; k-2, q}\)-minihyper, containing a (projected) Baer subgeometry \( PG(2\mu - 1, \sqrt{q})_i \) of \((F,w)\), passing through this particular (projected) Baer subgeometry \( PG(2\mu - 3, \sqrt{q}) \) of \( \Delta \) contained in \((F,w)\). These different (projected) Baer subgeometries \( PG(2\mu - 1, \sqrt{q})_i \), then define a (projected) Baer subgeometry \( PG(2\mu + 1, \sqrt{q}) \) contained in \((F,w)\).

But the dimension of these (projected) Baer subgeometries increases by two when going from the space \( \Delta' \) to the hyperplanes \( H_i \), and again the dimension of these (projected) Baer subgeometries increases by two when wishing to reconstruct the (projected) Baer subgeometry \( PG(2\mu + 1, \sqrt{q}) \), contained in \((F,w)\), defined by these (projected) Baer subgeometries \( PG(2\mu - 1, \sqrt{q})_i \). This increase by two in the dimension of these (projected) Baer subgeometries made it necessary to isolate in Subsection 2.2 a particular (projected) Baer subgeometry \( PG(2\mu - 2, \sqrt{q}) \), of minimal weight \( \alpha \), of \((F_H, w_H)\) in a \((k-3)\)-dimensional subspace \( \Delta \) of \( H \). This then led in many of the hyperplanes \( H_i \) through \( \Delta \) to (projected) Baer subgeometries \( PG(2\mu - 1, \sqrt{q})_i \) of \((F,w)\) through this isolated Baer subgeometry \( PG(2\mu - 2, \sqrt{q}) \) of \( \Delta \) in \((F,w)\). Using two of these (projected) Baer subgeometries \( PG(2\mu - 1, \sqrt{q})_i \) and \( PG(2\mu - 1, \sqrt{q})_j \), a (projected) Baer subgeometry \( PG(2\mu, \sqrt{q})_{ij} \) could be constructed, and these (projected) Baer subgeometries \( PG(2\mu, \sqrt{q})_{ij} \) then made it possible to reconstruct the (projected) Baer subgeometry \( PG(2\mu + 1, \sqrt{q}) \), of weight \( \alpha \), contained in \((F,w)\).

This approach enabled to construct the (projected) Baer subgeometries via a stepwise increase by one of the dimension, as was done for the reconstruction of the \( \mu \)-dimensional spaces \( PG(\mu, q) \) contained in \((F,w)\).

References

2. Beukemann, L., Metsch, K., Storme, L.: On weighted \({\delta v_\mu+1, \delta v_\mu; k-1, q}\)-minihypers, \( q \) square. In preparation.
On weighted \( \{\delta v_{\mu+1}, \delta v_{\mu}; k-1, q\} \)-minihypers, \( q \) square

On Bounds for Network Codes

Eimear Byrne

Abstract We give upper bounds on the size of an error correcting code for a data network. In our framework, the network has a single source, a collection of sink nodes $T$ and $n$ edges. A network code is a collection of codes $C := \{C_t \subset A_{n_t} : t \in T\}$ where $n_t$ is the number of edges incident with sink $t$ and $A$ is an $R$-$R$ bimodule for a finite ring $R$. For each $t \in T$, the ambient space $A_{n_t}$ of $C_t$ is $F_t(A_{n_t})$ for a left $R$-epimorphism $F_t$. Then $A_{n_t}$ is equipped with a weight function $w_t$ induced by a weight function $w$ on $A$ via $w_t(F_t(x)) = \min\{w(y) : x - y \in F_t\}$. We use the classical Plotkin and Elias bounds to derive upper bounds on the quantity $\min\{|C_t| : t \in T\}$ as we range over all such codes $C := \{C_t \subset A_{n_t} : t \in T\}$.

Keywords network code · network error-correction · coherent networks · Plotkin bound · Elias bound · finite Frobenius ring · homogeneous weight

1 Introduction

Coding in data communication networks has been shown to offer many advantages in terms of data rate, error correction and security. Many coding models have been considered for a variety of networks. For error-correcting network codes in multicast networks see [9,13–16].

We consider the following scenario: the network $\mathcal{N} = (V,E)$ is a directed acyclic graph with nodes/vertices $V$ and edge set $E$ of order $n$. We assume that $\mathcal{N}$ has a single source node $s$ incident with some $m$ outgoing edges and has several sinks labelled by elements of a set $T$. A message of $m$ packets is transmitted from the source node to be received at each sink in $T$. For each sink $t \in T$, let $E_t$ denote the set of edges

E. Byrne
School of Mathematical Sciences and Claude Shannon Institute,
University College Dublin, Ireland
E-mail: ebyrne@ucd.ie
incident with \( t \) and let \( |E_t| = n_t \). We furthermore assume that \( n_t \geq m \) for each \( t \) and indeed that there are at least \( m \) edge disjoint paths connecting \( s \) to \( t \). We will adopt a linear coding scheme, which means, as usual, that at each node in the network, linear combinations of packets on its incoming edges are transmitted along its outgoing edges.

Let \( R \) be a finite ring with unity and let \( \mathcal{A} \) be a finite \( R \)-\( R \) bimodule. The set of messages for \( \mathcal{N} \) is a subset \( \mathcal{M}_0 \) of \( \mathcal{A}^m \). Each message \( x_0 \in \mathcal{M}_0 \) corresponds to a unique network word \( x = [x_0, 0] \in \mathcal{M} := \{ [u, 0] : u \in \mathcal{M}_0 \} \subset \mathcal{A}^n \), canonically embedding \( \mathcal{A}^m \) into \( \mathcal{A}^n \). The network itself may be identified with \( \mathcal{A}^n \), where each \( i \)-th coordinate projection from \( \mathcal{A}^n \) onto \( \mathcal{A} \) corresponds to the \( i \)-th edge of the network, under some ordering.

A word \( z \in \mathcal{A}^n \) is transmitted along the network by an invertible transfer map

\[
\mathcal{F} : \mathcal{A}^n \longrightarrow \mathcal{A}^n : z \mapsto (f_1(z), \ldots, f_n(z)),
\]

for some \( R \)-linearly independent maps \( f_j \in \text{Hom}_R(\mathcal{A}^n, \mathcal{A}) \). If \( x \in \mathcal{A}^n \) is transmitted from the source node \( s \) and some edges of the network are corrupted by errors in the form of an error word \( e \in \mathcal{A}^n \) then the network transmission is given by \( y = \mathcal{F}(x + e) \). In other words, it is assumed that errors propagate through the network.

In the context of [9] (which is an error-free model) \( \mathcal{F} \) is represented by an invertible transfer matrix \( F \in R^{n \times n} \) with respect to some fixed basis. The network coding model described in [14–16] results from the case \( R = \mathcal{A} = GF(q) \). The approach in [13] corresponds to the case \( R = GF(q) \) and \( \mathcal{A} = GF(q)^n \).

For each \( t \in \mathcal{T} \), let \( \Pi_t : \mathcal{A}^n \longrightarrow \mathcal{A}^m : z \mapsto (z_j)_{j \in E_t} \) be the projection onto the coordinates indexed by the edges incident with \( t \) and define a map

\[
\mathcal{F}_t := \Pi_t \circ \mathcal{F} : \mathcal{A}^n \longrightarrow \mathcal{A}^m.
\]

In the multicast setting, for each \( t \in \mathcal{T} \), we require that \( \mathcal{F}_t : \mathcal{M} \longrightarrow \mathcal{A}^m \) be an injection, in order that each sink \( t \) can decode the transmitted word to the same unique message in \( \mathcal{M} \).

**Definition 1** Let \( \mathcal{N} \) be a network with transfer map \( \mathcal{F} \in \text{Aut}_R(\mathcal{A}^n) \). The network code for node \( t \) of \( (\mathcal{N}, \mathcal{F}) \) is the set

\[
\mathcal{C}_t := \{ \mathcal{F}_t(x) \in \mathcal{A}^m : x \in \mathcal{M} \} \subset \mathcal{A}^m.
\]

The network code of \( (\mathcal{N}, \mathcal{F}) \) is the collection \( \mathcal{C} := \{ \mathcal{C}_t : t \in \mathcal{T} \} \).

Note that neither \( \mathcal{M} \) nor any \( \mathcal{C}_t \) need be \( R \)-linear; the linearity of \( \mathcal{C} \) refers only to linearity of the transfer map \( \mathcal{F} \).

The received word at node \( t \) is given by \( y_t = \mathcal{F}_t(x + e) \). We denote by \( \mathcal{K}_t \) the kernel of the map \( \mathcal{F}_t \) in \( \mathcal{A}^n \) so that \( \mathcal{C}_t = \mathcal{F}_t(\mathcal{M}) \subset \mathcal{A}^m \cong \mathcal{A}^n / \mathcal{K}_t \). Observe that if \( e \in \mathcal{K}_t \) then \( y_t = \mathcal{F}_t(x + e) = \mathcal{F}_t(x) \) is received as if no errors have occurred. If \( m = n_t \), the kernel \( \mathcal{K}_t \) is trivial and the decoder will not detect any errors.

A weight function \( w \) on \( \mathcal{A}^m \) induces a weight \( w_0 \) on \( \mathcal{A}^n \) as follows:

\[
w_t(u) := w(x + \mathcal{K}_t) = \min \{ w(z) : \mathcal{F}_t(z) = u \},\]
for some $x$ in the preimage of $u$ under $\mathcal{F}_t$. If $w$ determines a distance function $d$ on $\mathcal{A}^n$ by $d(x, y) := w(x - y)$, then $d_t(u, v) := w_t(u - v)$ is also a distance function on $\mathcal{A}^m$.

Given the received word $y_t$, the decoder at node $t$ decides that $c = \mathcal{F}_t(x)$ has been transmitted if $d_t(y_t, c) < d_t(y_t, c')$ for all $c' \in \mathcal{C}_t$.

**Example 1** Let $R = \mathcal{A} = GF(q)$ and let $w$ denote the usual Hamming weight on $GF(q)^n$. Let $\mathcal{N}$ be a network and let $\mathcal{C}_t$ be a network code for $\mathcal{N}$ at one of its sink nodes $t$. $\mathcal{F}_t$ has a representation as an $n \times n_t$ matrix $F_t$ with respect to a chosen basis. Then $w_t(u) = \min \{w(z) : zF_t = u, \} = w(x + \mathcal{K}_t)$ counts the minimum number of linearly independent rows of $F_t$ required to obtain a representation of $u = xF_t$. If $x \in \mathcal{M}$ is transmitted and $y$ is received at $t$, the decoder will decode to $y = eF_t \in \mathcal{C}_t$ for some error $e \in GF(q)^n$ of least Hamming weight satisfying $y = (x + e)F_t$. In other words the decoder assumes that the least number of edges resulting in a non-trivial contribution to the computation of $y$ have been affected by noise during transmission.

We denote by $d_t$ the minimum distance of $\mathcal{C}_t$ with respect to $d_t$. We write $\ell_t$ to denote the size of the support of $\mathcal{K}_t$. We write $\mathcal{M}_t := \mathcal{F}_t^{-1}(\mathcal{C}_t)$ to denote the preimage of $\mathcal{C}_t$ in $\mathcal{A}^n$ under $\mathcal{F}_t$. Then $|\mathcal{M}_t| = |\mathcal{C}_t| |\mathcal{K}_t|$.

We say that $\mathcal{C}$ is an $(n, \{n_t, \ell_t, |\mathcal{K}_t|, d_t\} : t \in \mathcal{T})$ network code. We define $s(\mathcal{C}) := \min \{|\mathcal{C}_t| : t \in \mathcal{T}\}$, which we call the size of $\mathcal{C}$, and seek upper bounds on this number, which is the effective maximum possible size of the message space $\mathcal{M}$.

**Definition 2** We denote by $A(n, \{n_t, \ell_t, d_t\} : t \in \mathcal{T})$ the maximum size $s(\mathcal{C})$ of any $(n, \{n_t, \ell_t, |\mathcal{K}_t|, d_t\} : t \in \mathcal{T})$ network code $\mathcal{C}$.

**Modules and Homogeneous Weights**

The homogeneous weight was first introduced on the ring $\mathbb{Z}_m$ in [2]. Generalizations of this weight function have appeared in [3, 4, 7]. For coding theoretic purposes, these are often best defined on a Frobenius bimodule.

We define a weight function, or weight on an $R$-module $M$ to be a map $w : M \rightarrow \mathbb{R}$ such that $w(0) = 0$. The homogeneity conditions in [3] are given by the following.

**Definition 3** A weight function $w$ on a left $R$-module $M$ is called (left) homogeneous if

1. **H1** If $Rx = Ry$ then $w(x) = w(y)$ for all $x, y \in M$.
2. **H2** There exists a real number $\gamma$ such that
   \[
   \sum_{y \in Rx} w(y) = \gamma |Rx| \quad \forall \ 0 \neq x \in M.
   \]

Right homogeneous weights are defined similarly. In [7] condition H2 is given as:

1. **H2’** There exists a real number $\gamma$ such that
   \[
   \sum_{x \in V} w(x) = \gamma |V| \quad \forall \ 0 \neq V <_R M.
   \]
In fact, if \(RM\) is a left Frobenius module, in particular if its socle is cyclic, then \(H2\) implies \(H2'\). Moreover, a homogeneous weight function exists on any finite module \(RM\) and is unique up to choice of average weight \(\gamma\) [7].

**Example 2** For the case \(R = M = GF(q)\), the Hamming weight is homogeneous with average weight \(\gamma = \frac{q-1}{q}\). For the case \(R = M = \mathbb{Z}_4\), the Lee weight is homogeneous with average weight \(\gamma = 1\).

**Example 3** Let \(R = M = GF(q)^{2\times 2}\). Then it is straightforward to check that the weight function \(w\) on \(M\) defined by

\[
w(x) = \begin{cases} 
\frac{q^3 - q^2 - q}{q^3 - q^2 - q + 1} & \text{if } \text{rank}_{GF(q)}(x) = 2, \\
\frac{q^2}{q^3 - 1} & \text{if } \text{rank}_{GF(q)}(x) = 1, \\
0 & \text{if } x = 0,
\end{cases}
\]

is homogeneous with average value \(\gamma = 1\).

For the case \(R = GF(q), M = GF(q)^{2\times 2}\) only the Hamming weight, for \(\gamma = \frac{q-1}{q}\), is homogeneous.

We extend \(w\) to a weight function on \(\mathcal{A}^n\) in the obvious way:

\[
w : \mathcal{A}^n \rightarrow \mathbb{R} : (c_1, \ldots, c_n) \mapsto \sum_{i=1}^{n} w(c_i).
\]

In [3], the authors show that every finite unital ring \(R\) has a quasi-Frobenius bimodule, which is unique up to right and left \(R\)-isomorphism if its socle is cyclic both as a left and right \(R\)-module. Such a module is then called a Frobenius bimodule.

Let \(A := \text{Hom}_Z(\mathcal{A}, C^n)\), the group of characters of the additive group of \(\mathcal{A}\). \(\hat{A}\) is an \(R\)-\(R\) bimodule according to the relations

\[
'\chi(x) = \chi(rx), \quad \chi'(x) = \chi(xr)
\]

for all \(r \in R, x \in \mathcal{A}\) and \(\chi \in \hat{A}\). A character \(\chi \in \hat{A}\) is called (left) generating if given any \(\phi \in \hat{R}\) there is some \(r \in R\) satisfying \(\phi = '\chi\). This is equivalent to the property that \(\ker \chi\) contains no left \(R\)-submodule of \(\mathcal{A}\).

**Definition 4** The bimodule \(R \mathcal{A} R\) is a Frobenius bimodule if

\[
R \mathcal{A} \cong R \hat{A} \quad \text{and} \quad \hat{A} R \cong \hat{R} R.
\]

By duality, if \(R \mathcal{A} R\) is Frobenius then \(R \hat{A} \cong \hat{R} R\) and \(\hat{A} R \cong \hat{R} R\). In particular, if \(R \mathcal{A} R\) is a Frobenius bimodule then \(\mathcal{A}\) is generated by a character \(\chi\) both as a left and as a right \(R\)-module.

The existence of such a generating character \(\chi \in \mathcal{A}\) gives the following characterization of the homogeneous weight on a Frobenius bimodule (cf [6]). The proof is straightforward.
Lemma 1 Let \( \mathcal{A}_R \) be a Frobenius bimodule with generating character \( \chi \). Then the weight function

\[
w : \mathcal{A} \longrightarrow \mathbb{R} : a \mapsto \gamma \left( 1 - \frac{1}{|R|^2} \sum_{u \in R} \chi(au) \right)
\]
is homogeneous.

For a positive integer \( k \), word \( z \in \mathcal{A}^k \) and set \( X \subset \{1, \ldots, k\} \) we define \( \pi_X(z) \). Given an \( R \)-submodule \( M < \mathcal{A}^k \), we write \( \text{supp}(M) \) to denote the set \( \{i : \pi_i(c) \neq 0 \text{ for some } c \in M\} \) for each \( i \).

Using the character-theoretic description of the homogeneous weight given above, the following result can be shown, with the proof proceeding almost exactly as in [1, Lemma 1].

Lemma 2 Let \( \mathcal{A}_R \) be a Frobenius bimodule with homogenous weight function \( w : \mathcal{A} \longrightarrow \mathbb{R} \). Let \( M \) be an \( R \)-submodule of \( \mathcal{A}^n \) and let \( x \in \mathcal{A}^n \). Then

\[
\sum_{c \in M} w(x + c) = \gamma |M||\text{supp}(M)| + |M|w(\pi_{\text{supp}(M)}(x)).
\]

Unless stated otherwise, for the remainder we assume that \( \mathcal{A} \) is a Frobenius bimodule and that for each linear map \( \mathcal{F}_t : \mathcal{A}^n \longrightarrow \mathcal{A}^m \) the weight function \( w_t \) is induced by the homogeneous weight \( w \) on \( \mathcal{A} \), extended to a weight on \( \mathcal{A}^n \).

2 A Plotkin-Like Bound

We seek an upper bound on \( |C_t| \) for each \( (n_t, \ell_t, |C_t|, d_t) \) code \( C_t \). Following the usual argument for the classical Plotkin bound, we obtain lower and upper bounds on \( \sum_{c, c' \in C_t} d_t(c, c') \). Proposition 2.1 of [5] gives a Plotkin bound for codes over finite Frobenius rings with respect to the homogeneous weight, which has the trivial extension:

Lemma 3 Let \( C \subset \mathcal{A}^n \) have minimum homogeneous distance \( d(C) \). Then

\[
|C|(|C| - 1)d(C) \leq \sum_{x, y \in C} w(x - y) \leq \gamma n|C|.
\]

Theorem 1 Let \( C_t \subset \mathcal{A}^n \) be the \( (n_t, |C_t|, d_t) \) network code for node \( t \), let \( \mathcal{M}_t = \mathcal{F}_t^{-1}(C_t) \) and let \( s_t := |\text{supp}(\mathcal{M}_t)| \). If \( d_t > \gamma s_t \) then

\[
|C_t| \leq \frac{d_t - \gamma s_t}{d_t - \gamma s_t}.
\]
Proof We give an estimate of the sum of the distances between ordered pairs of distinct codewords of $C_t$:

$$|C_t||C_t| - 1) d_t \leq \sum_{F_t(x), F_t(y) \in C_t} w_\gamma(F_t(x) - F_t(y)),$$

$$= \sum_{F_t(x), F_t(y) \in C_t} w(x - y + K_t),$$

$$\leq \sum_{F_t(x), F_t(y) \in C_t} \frac{1}{|K_t|} \sum_{k \in K_t} w(x - y + k),$$

$$= \frac{1}{|K_t|^2} \sum_{x, y \in M_t; x - y \in K_t} w(x - y),$$

$$= \frac{1}{|K_t|^2} \left( \sum_{x, y \in M_t} w(x - y) - \sum_{x, y \in M_t; x - y \in K_t} w(x - y) \right),$$

$$= \frac{1}{|K_t|^2} \left( \sum_{x, y \in M_t} w(x - y) - |C_t||K_t| \sum_{z \in K_t} w(z) \right),$$

$$\leq |C_t|^2 s_t \gamma - |C_t||K_t| \gamma,$$

from Lemmas 2 and 3. Now rearrange to obtain

$$|C_t| \leq \frac{d_t - \gamma \ell_t}{d_t - \gamma s_t},$$

as long as $d_t > \gamma s_t$.

We remark that if $\ell_t = s_t$, then the inequality $(|C_t| - 1) d_t \leq |C_t| s_t \gamma - \ell_t \gamma$ implies that $d_t \leq s_t \gamma$. If $K_t$ is trivial then $\ell_t = 0$ and Theorem 1 is the classical Plotkin bound [5, Theorem 2.2].

Corollary 1 Let $d = \min\{d_t : t \in T\} > \gamma n$ and let $\ell = \min\{\ell_t : t \in T\}$. Then

$$A(n, \{(n_t, \ell_t, d_t) : t \in T\}) \leq \min \left\{ \frac{d_t - \gamma \ell_t}{d_t - \gamma n} : t \in T \right\} \leq \frac{d - \gamma \ell}{d - \gamma n}.$$

Observe that the upper bound in Corollary 1 depends on $n, \ell_t$ and $d_t$, but not on $n_t$. However, since $\ell_t \geq n - n_t$ we immediately obtain the following.

Corollary 2 Let $d = \min\{d_t : t \in T\} > \gamma n$. Then

$$A(n, \{(n_t, \ell_t, d_t) : t \in T\}) \leq \min \left\{ 1 + \frac{\gamma n_t}{d_t - \gamma n} : t \in T \right\}.$$
3 An Elias-Like Bound

We recall the following well-known lemma (see, for example [10, Lemma 5.2.9]).

**Lemma 4** Let $A, B \subseteq \mathcal{A}^N$. Then there exists $x \in \mathcal{A}^N$ such that

$$|B| \leq \frac{|(x+A) \cap B|}{|A|}.$$  

For each nonnegative real number $r$ we define

$$B_i^r(r) := \{z \in \mathcal{A}^m : \frac{1}{|A_i|} \sum_{i \in \mathcal{A}_i(z) = z} w(x) \leq r\}.$$  

It is easy to deduce from this lemma that

$$|\mathcal{C}_i| \leq \frac{|B_i^r(r) \cap \mathcal{C}_i|}{|B_i^r(r)|}.$$  

To simplify notation, for each $t \in \mathcal{T}$ and vector $x \in \mathcal{A}^m$, we write

$$\pi_t(x) := (x_i)_{i \in \text{supp}, x_i} \in \mathcal{A}^{n-t_i}$$

and $\pi'_t(x) := (x_i)_{i \in \text{supp}, x_i} \in \mathcal{A}^{t_i}$.

Note that if $\mathcal{F}_i(x) = \mathcal{F}_i(y)$ for some $x, y \in \mathcal{A}^m$ then $\pi_t(x) = \pi_t(y)$. We will use this elementary fact throughout.

**Corollary 3** Let $r \geq \ell_t \gamma_t$. Then

$$B_i^r(r) = \{\mathcal{F}_i(x) \in \mathcal{A}^m : w(\pi_t(x)) \leq r - \ell_t \gamma_t\}.$$  

**Proof** Let $z \in \mathcal{A}^m$. Applying Lemma 2, $z \in B_i^r(r)$ if and only if

$$r \geq \frac{1}{|A_i|} \sum_{i \in \mathcal{A}_i(z) = z} w(x) = \gamma t_i + w(\pi_t(x)).$$

\[ \square \]

**Corollary 4** Let $r \geq \gamma t_i$. Then

$$|B_i^r(r)| = |B_i^{n-t_i}(r - \gamma t_i)||\mathcal{A}^{|t_i-n+m|}.$$  

**Proof** Let $x \in \mathcal{A}^m$. From Corollary 3, $\mathcal{F}_i(x) \in \mathcal{B}_i^r(r)$ if and only if $\pi_t(x) \in \mathcal{B}_i^{n-t_i}(r - \gamma t_i)$. Let $\mathcal{X}_i := \{\pi_t(z) : z \in \mathcal{X}_i\} \subseteq \mathcal{A}^{|t_i|}$. Clearly $|\mathcal{X}_i| = |\mathcal{X}_t|$, so there are exactly $|\mathcal{A}^{|t_i-n+m|}$ distinct cosets of $\mathcal{X}_i$ in $\mathcal{A}^{|t_i|}$. Then for each $u \in \mathcal{B}_i^{n-t_i}(r - \gamma t_i)$, there are exactly $|\mathcal{A}^{|t_i-n+m|}$ distinct elements $\mathcal{F}_i(x) \in \mathcal{A}^m$ satisfying $\pi_t(x) = u$. \[ \square \]

The following is now immediate.

**Theorem 2** Let $r - \gamma t_i > 0$ for each $t \in \mathcal{T}$. Then

$$A(n, \{ (n_t, \ell_t, d_t) : t \in \mathcal{T} \}) \leq \min \left\{ \frac{|B_i^n(r) \cap \mathcal{C}_i|}{|B_i^{n-t_i}(r - \gamma t_i)|} : t \in \mathcal{T} \right\}.$$  

(4)
We now obtain an upper bound on the size of $B_i(r) := B_{n, t}^\alpha \cap \mathcal{C}_i$. Clearly, $B_i(r)$ has minimum distance $d_i(\mathcal{B}_i(r)) \geq d_i$.

For each $t \in \mathcal{T}$, let $\{x_1, \ldots, x_{|B_i(r)|}\}$ be a set of distinct coset representatives of $\mathcal{K}_i$ in $\mathcal{T}_{-1}^{-1}(B_i(r))$. We denote by $B_{n, t}^\alpha$ the multiset of words in $\mathcal{X}^{n-t_i}$ constructed as

$$B_{n, t}^\alpha := \{ \pi_0(x_1), \ldots, \pi_0(x_{|B_i(r)|}) \},$$

where $|B_{n, t}^\alpha| = |B_i(r)|$.

**Theorem 3** Let $r \geq \gamma t_i$. Then

$$|B_i(r)|(\gamma t_i) \geq |B_i(r)| - 1) \gamma t_i \leq \sum_{u,v \in B_{n, t}^\alpha} w(u - v).$$

**Proof** Using Lemma 2, we obtain

$$|B_i(r)|(\gamma t_i) \geq (|B_i(r)| - 1) \gamma t_i = \sum_{\mathcal{F}_t(x), \mathcal{F}_t(y) \in B_i(r)} w_t(\mathcal{F}_t(x) - \mathcal{F}_t(y)) \leq |B_i(r)|(\gamma t_i) \gamma t_i + \sum_{\mathcal{F}_t(x), \mathcal{F}_t(y) \in B_i(r)} w_t(\mathcal{F}_t(x) - \mathcal{F}_t(y)) = |B_i(r)|(\gamma t_i) \gamma t_i + \sum_{u,v \in B_{n, t}^\alpha} w(u - v).$$

**Lemma 5** Let $r \geq \gamma t_i$. Then $d_{\gamma t_i}(B_{n, t}^\alpha) \geq d_i - \gamma t_i$.

**Proof** As before, let $\mathcal{K}_i' := \{ \pi_0(z) : z \in \mathcal{K}_i \} \subset \mathcal{X}^t$. Let $u, v \in B_{n, t}^\alpha$. Let $x,y \in \mathcal{T}_{-1}(B_i(r))$ satisfy $\pi_0(x) = u, \pi_0(y) = v$, and

$$d_i \leq d_t(\mathcal{F}_t(x), \mathcal{F}_t(y)) = w_t(x - y + \mathcal{K}_i) = w_t(\pi_0(x) - \pi_0(y) + \mathcal{K}_i) + w_t(\pi_0(x) - \pi_0(y)) \leq \gamma t_i + w_t(u - v).$$

In particular $d_t(\mathcal{F}_t(x), \mathcal{F}_t(y)) \geq d_i - \gamma t_i$. □

We have the following result, given in the proof of [5, Corollary 3.3].

**Lemma 6** Let $s \leq \gamma N$ and let $W \subset \mathcal{X}^N$. If $w(v) \leq s$ for every $v \in W$ then

$$\sum_{u,v \in W} w(u - v) \leq 2|W|^2 s - |W|^2 s^2 \frac{s^2}{\gamma N}.$$

Observing that the above result is valid for a multiset of words $W$, and then substituting $N = n - t_i$, $s = r - \gamma t_i$, $W = B_{n, t}^\alpha$, and $d_t(\mathcal{F}_t(x), \mathcal{F}_t(y)) \geq d_i - \gamma t_i$ immediately gives:
Corollary 5 Let \( r \leq \gamma n \) and let \( n > \ell_t \). Then

\[
|B_t(r)|(|B_t(r)| - 1)(d_t - \gamma \ell_t) \leq 2|B_t(r)|^2(r - \gamma \ell_t) - |B_t(r)|^2 \frac{(r - \gamma \ell_t)^2}{\gamma(n - \ell_t)}.
\]

Moreover, if \( r \leq \gamma n - \sqrt{\gamma(n - \gamma \ell_t)(\gamma n - d_t)} \) then

\[
|B_t(r)| \leq f(r, n, \ell_t, d_t) := \frac{\gamma(d_t - \gamma \ell_t)(n - \ell_t)}{(r - \gamma n)^2 - \gamma(n - \gamma \ell_t)(n - \gamma \ell_t)}.
\]

Theorem 4 Let \( d_t \leq \gamma n \) and let \( \gamma \ell_t \leq r \leq \gamma n - \sqrt{\gamma(n - d_t)(n - \ell_t)} \). Then

\[
\|\gamma_t\| \leq \frac{f(r, n, \ell_t, d_t)}{|B^{n-t}(r - \gamma \ell_t)|}.
\]

In particular if the above hypothesis holds for each \( t \in \mathcal{T} \), then

\[
A(n, \{(n, \ell_t, d_t) : t \in \mathcal{T}\}) \leq \min \left\{ \frac{f(r, n, \ell_t, d_t)}{|B^{n-t}(r - \gamma \ell_t)|} : t \in \mathcal{T} \right\}.
\]

4 Asymptotic Bounds

Asymptotic versions of these bounds are expressed by finding upper bounds on:

\[
\alpha_{\mathcal{A}'}(\{(v_t, \lambda_t, \delta_t) : t \in \mathcal{T}\}) := \lim sup n^{-1} \log_{|\mathcal{A}'|} \left( A(n, \{(v_t n, \lambda_n, \delta_n) : t \in \mathcal{T}\}) \right).
\]

As in the classical Hamming case, we will require an asymptotic expression for the size of the homogeneous sphere \( B^N(\delta N) \subset \mathcal{A}'^N \). This was essentially answered first in [11] and in a slightly different form (which we use here) in [5, Theorem 4.1] as follows:

Lemma 7 For all \( \delta \in [0, \gamma] \) there holds:

\[
\lim_{N \to \infty} N^{-1} \log_{|\mathcal{A}'|} |B^N(\delta N)| = \min \left\{ \log_{|\mathcal{A}'|} \left( \sum_{a \in \mathcal{A}'} Z^{n(a)-\delta} \right) : Z \in [0, 1] \right\}.
\]

Definition 5 Let \( \delta \geq 0 \). We define the function

\[
H_{\mathcal{A}'}(\delta) := \min \left\{ \log_{|\mathcal{A}'|} \left( \sum_{a \in \mathcal{A}'} Z^{n(a)-\delta} \right) : Z \in [0, 1] \right\}.
\]

4.1 An Asymptotic Plotkin Bound

Theorem 5 \( \alpha_{\mathcal{A}'}(\{(v_t, \lambda_t, \delta_t) : t \in \mathcal{T}\}) \leq \left\{\begin{array}{ll}0 & \text{if } \delta > \gamma \\ 1 - \frac{\delta}{\gamma} & \text{if } \delta \leq \gamma\end{array}\right. \)

where \( \delta = \min\{\delta_t : t \in \mathcal{T}\} \).
4.2 An Asymptotic Elias Bound

**Theorem 6** Let \( \mathcal{T} \) be a non-empty set and let \( \rho > 0 \). For each \( t \in \mathcal{T} \) let \( \nu_t, \lambda_t, \delta_t \in (0, 1) \) satisfy \( \delta_t \leq \gamma, \gamma \lambda_t \leq \rho \leq \gamma - \sqrt{\gamma(\gamma - \delta_t)(1 - \lambda_t)} \). For each positive integer \( n \) define

\[
h_n(u, v) := \frac{f(\rho, 1, u, v)}{|A|n^{1-u}},
\]

\[
(\lambda, \delta) := \arg\min\{h_n(\lambda_t, \delta_t): t \in \mathcal{T}\}
\]

and \( \xi := \frac{\rho - \gamma \lambda}{1 - \lambda} \). Then

\[
\lim_{n \to \infty} n^{-1} \log_{|\mathcal{A}|} (\lambda(n, \{(\nu_t n, \lambda_t n, \delta_t n): t \in \mathcal{T}\})) \leq 1 - \lambda - H_{\mathcal{A}}(\xi).
\]

In particular,

\[
\alpha_{\mathcal{A}}(\{(\nu_t, \delta_t, \lambda_t): t \in \mathcal{T}\}) \leq 1 - \lambda - H_{\mathcal{A}}\left(\gamma - \sqrt{\frac{\gamma(\gamma - \delta)}{1 - \lambda}}\right).
\]

As \( \lambda \to 0 \), the upper bound on \( \alpha_{\mathcal{A}}(\{(\nu_t, \delta_t, \lambda_t): t \in \mathcal{T}\}) \) given above becomes the asymptotic Elias bound of [5].

**References**

Tail-Biting Trellis Realizations and Local Reductions

Heide Gluesing-Lueerssen

Received: date / Accepted: date

Abstract. This paper investigates tail-biting trellis realizations for linear block codes. By considering the trellis and its dual simultaneously, a characterization for irreducibility on given intervals of the time axis is given, and a constructive reduction procedure is presented. For fragments of length less than the minimum span length of the code, the criterion amounts to fragment observability and fragment controllability.

Keywords Codes on Graphs · Minimality · Reducibility

Mathematics Subject Classification (2000) 94B05 · 93B20 · 93B05

1 Introduction

Tail-biting trellis realizations form a particular type of realizations of codes on graphs. The latter have gained a lot of attention because they give rise to powerful decoding algorithms; see [14, 19, 11, 13, 18, 9, 8, 2] for tail-biting trellises and [4, 15, 10, 1] for normal realizations on general graphs. While realizations on cycle-free graphs are by now fully understood, many questions remain to be answered for graphs with cycles. Tail-biting trellises form the simplest such case because the graph consists of a single cycle.

Performing effective decoding algorithms requires efficient realizations, that is, small realizations with respect to some complexity measure. Minimal re-
alization theory, however, turns out to be much more challenging for graphs with cycles than for the cycle-free case.

In the cycle-free case, minimality (with respect to state space dimensions) can be characterized intrinsically, and all minimal realizations of a given code on the same cycle-free graph are isomorphic. Moreover, any nonminimal realizations can be reduced in a constructive way. This is classical material for conventional trellises (where the graph is a path), see [3,17,16,20], and has been generalized to arbitrary cycle-free graphs in [6]; see also [10].

For realizations on graphs with cycles no such characterizations and reductions are known. A systematic study of the simplest case, namely tail-biting trellises, was initiated by Koetter/Vardy [13]. Among other things, they highlighted the problem that all meaningful concepts of complexity measures lead to different notions of minimality. As a result, a given code does not have a unique minimal tail-biting trellis. Furthermore, no intrinsic characterization of minimality and no reduction procedure resulting in a minimal trellis are known.

The goal of this paper is to develop constructive methods of reducing the complexity of a given tail-biting trellis and present irreducibility criteria. They are given in terms of intrinsic, easy-to-check trellis properties and characterize irreducibility on intervals of the circular time axis up to a certain length. Our approach provides a toolbox of reduction methods, which we believe can be exploited further and will ultimately lead to minimal trellises. This is made precise in a final conjecture.

2 Basic Notions

Throughout, a linear block code $C$ over a finite field $F$ is a subspace of a symbol configuration space $A = \prod_{i=0}^{m-1} A_i$, where each symbol space $A_i$ is a finite-dimensional vector space over $F$.

A linear tail-biting trellis realization $R$ of length $m$ is a normal linear realization, in the sense of [4] or [6], on a graph that is a single cycle of length $m$. It thus consists of the spaces $A_i$, a set of state spaces $S_i$, and a set of constraint codes $C_i \subseteq S_i \times A_i \times S_i+1$ for $i \in \mathbb{Z}_m$. Every constraint code thus involves precisely two state variables. All variable alphabets are finite-dimensional vector spaces over $F$. The elements of $C_i$ are called constraint codewords or branches.

We will call a linear tail-biting trellis realization simply a trellis.

The behavior of $R$ is the set $\mathcal{B} := \{(a,s) \in A \times S \mid (s_i, a_i, s_{i+1}) \in C_i \text{ for all } i \in \mathbb{Z}_m\}$, where $S := \prod_{i=0}^{m-1} S_i$ is the state configuration space. The pairs $(a,s) \in \mathcal{B}$ are the valid trajectories. The code $C$ realized by the trellis is the projection $C := \mathcal{B}|_A := \{a \in A \mid (a,s) \in \mathcal{B} \text{ for some } s \in S\}$.

The dual trellis of $R$, denoted by $R^\circ$, is defined as the trellis where the symbol and state spaces are replaced by their linear algebra duals $\hat{A}_i, \hat{S}_i$, and where the constraint codes $C_i$ are replaced by $C_i^\circ := \{\langle \hat{s}_i, \hat{a}_i, \hat{s}_{i+1} \rangle \mid \langle \hat{s}_i, s_i \rangle + \langle \hat{a}_i, a_i \rangle - \langle \hat{s}_{i+1}, s_{i+1} \rangle = 0 \text{ for all } (s_i, a_i, s_{i+1}) \in C_i\}$. Again, the behavior of $R^\circ$ is $\mathcal{B}^\circ = \{(\hat{a}, \hat{s}) \mid (\hat{s}_i, \hat{a}_i, \hat{s}_{i+1}) \in C_i^\circ \text{ for all } i\}$. The
Normal Realization Duality Theorem [4,15,1,5,6] states that if \( R \) realizes \( C \), then its dual \( R^\circ \) realizes the dual code \( C^\perp := \{ \hat{a} \in \hat{A} \mid \langle \hat{a}, a \rangle = 0 \text{ for all } a \in C \} \). In other words, \( (B^\circ)^\perp = C^\perp \).

The proof of the duality theorem makes use of the following fact, which will also play a crucial role in this paper. Let \( C \) be a subspace of a vector space \( T = T_1 \times T_2 \). Projection and cross-section of \( C \) on \( T_1 \) are defined as \( C_{|T_1} := \{ t_1 \mid \exists t_2 \in T_2 : (t_1, t_2) \in C \} \) and \( C_{T_1} := \{ t_1 \mid (t_1, 0) \in C \} \), respectively.

Given inner products between \( T_i \) and \( \hat{T}_j \), extended naturally to \( T_1 \times T_2 \) and its dual \( \hat{T}_1 \times \hat{T}_2 \), projection/cross-section duality [4] (see also [6, Sec. II.H]) states that
\[
(C_{|T_1})^\perp = (C^\perp)_{T_1}
\]
for any subspace \( C \subseteq T_1 \times T_2 \) and its orthogonal subspace \( C^\perp \subseteq \hat{T}_1 \times \hat{T}_2 \).

The following two examples will be used as running examples throughout this paper. In both cases the ground field is \( F_2 \). The dashed (resp. solid) lines denote branches with symbol variable 0 (resp. 1). We also show a choice of state labels. The choice does not play any particular role and may be changed at any time index using a state space isomorphism. The state space at time 0 appears at both ends of the trellises. Note that valid configurations correspond to cycles in the trellis; in other words, the valid configurations are exactly the paths in the trellis that start and end at the same state at time 0. The dual state space will always be identical to the primal space, and the inner product is the one induced by the standard dot product on \( F_2 \).

For both examples, the given trellises are mutually dual.

![Fig. 1](image1.png)  
Fig. 1 Trellises representing the code \( C = (110, 101) \) and \( C^\perp = (111) \), respectively.

![Fig. 2](image2.png)  
Fig. 2 Trellises representing the code \( C = (01110, 10010, 01101) \) and \( C^\perp = (10111, 01100) \).

Definition 1 will be the central concept of this paper. We use the following interval notation. For \( j, k \in \mathbb{Z}_m \) and \( j \neq k \) let \( [j, k) := \{ j, j+1, \ldots, k-1 \} \subseteq \mathbb{Z}_m \) denote a (possibly circular) subinterval of \( \mathbb{Z}_m \). Hence \([j, k) \cup [k, j) = \mathbb{Z}_m \).

**Definition 1** A \((j,k)\)-reduction \( \hat{R} \) of a trellis \( R \) is a replacement of the state spaces \( S_{j+1}, S_{j+2}, \ldots, S_{k-1} \) by state spaces \( \hat{S}_{j+1}, \hat{S}_{j+2}, \ldots, \hat{S}_{k-1} \) of at most
the same size and of the adjacent constraint codes \( C_j, \ldots, C_{k-1} \) by suitable constraint codes \( \tilde{C}_j, \ldots, \tilde{C}_{k-1} \) of any size, without changing the rest of the realization or the code \( C \) that it realizes. We also call this a \( t \)-\emph{reduction}, where \( t = (k - j) \mod m \in \{1, \ldots, m - 1\} \). The reduction will be called \( \emph{strict} \) if at least one of the state space sizes decreases strictly. A trellis \( R \) is called \( t\)-irreducible if each \( t \)-reduction is isomorphic to \( R \).

Throughout this paper we will not be concerned with the constraint code dimensions. It turns out that for the procedures given in this paper, no constraint code dimension will increase. Details can be found in [7].

Evidently a trellis \( R \) is (strictly) \( t \)-reducible if and only if the dual trellis \( R^\circ \) is (strictly) \( t \)-reducible.

As we will see, non-strict \( t \)-reductions form a useful concept in our considerations. They are needed to derive strictly reducible trellises so that ultimately a net reduction in state space sizes is achieved. For instance, 1-reductions are non-strict as they consist of the replacement of a single constraint code (e.g., deleting or adding branches).

The most important examples of strict 2-reductions are the mutually dual processes of trimming and merging. Trimming a trellis simply means to restrict one state space to a subspace, while merging means to replace a state space by a quotient space. In both cases the two adjacent constraint codes are adjusted in the obvious way. Projection/cross-section duality implies that the trellis \( R' \) is obtained from \( R \) by trimming a state space \( S_j \) to a subspace \( Y_j \) if and only if \( (R')^\circ \) is obtained from \( R^\circ \) by merging \( 
\hat{\hat{S}}_j \) to \( \hat{\hat{S}}_j/Y_j \). If the code realized by the trellis does not change, then trimming and merging obviously form simultaneous strict 2-reductions of the trellis and its dual in the sense of Def. 1.

We say a trellis is \emph{nonmergeable} (resp. \emph{non-trimmable}) if no state space can be merged (resp. trimmed) without changing the code realized by the trellis. By the above, non-mergeability and non-trimmability are mutually dual.

A trellis is \emph{state-trim} if each state appears on a valid trajectory, i.e., \( B|_{S_i} = S_i \) for all \( i \). It is called \emph{branch-trim} if each branch appears on a valid trajectory, i.e., \( B|_{S_i \times A_i \times S_{i+1}} = C_i \) for all \( i \).

A trellis \( R \) for the code \( C \) is called \emph{observable} if for each \( a \in C \) there is precisely one valid trajectory \((a, s) \in B \). A trellis is called \emph{controllable} if its dual is observable. For details on these notions and a simple controllability test we refer to [6, Sec. IV-C].

A trellis is called \emph{trim at} \( S_i \) if \( (C_i-1)|_{S_i} = S_i = (\tilde{C}_i)|_{S_i} \), and \emph{proper at} \( S_i \) if \( (C_i-1)|_{S_i} = \{0\} = (\tilde{C}_i)|_{S_i} \). The former means that each state in \( S_i \) has an incoming branch and an outgoing branch while the latter means that there are no nontrivial branches of the form \((0,0,s_i)\) in \( \tilde{C}_i \) and none of the form \((s_i,0,0)\) in \( C_i \). We call a trellis \emph{trim} (resp. \emph{proper}) if it is trim (resp. proper) at each state space. Using projection/cross-section duality, one obtains immediately that \( R \) is trim at \( S_i \) if and only if \( R^\circ \) is proper at \( \hat{\hat{S}}_i \); see [8,6].

Let us discuss all these properties for the examples in Fig. 1 and 2. It is easy to see that both trellises in Fig. 1 are trim, proper, observable, and controllable (TPOC, for short). The second trellis is not state-trim and not branch-trim,
and in fact, can be trimmed at time 2. By duality, the first trellis is mergeable, and thus both are strictly 2-reducible. Carrying out these reductions leads to the following dual pair (which can be further trimmed/merged).

![Fig. 3 State-merged trellis and state-trimmed dual trellis resulting from Fig. 1](image)

The trellises in Fig. 2 are both TPOC, state-trim and non-trimmable (all of which can easily be verified), and thus non-mergeable, by duality. Only the first trellis is branch-trim. Trimming the unused branches of the second trellis and the implications thereof will be discussed after Lemma 2.

We close this section with a list of necessary conditions for a trellis to be irreducible [6,7]. Most of them follow easily by using trimming and merging.

**Theorem 1** Any 2-irreducible trellis is TPOC, state-trim, and nonmergeable.

Unobservable trellises can be reduced as follows.

**Remark 1** An unobservable trellis can be trimmed to any subspace $T_i$ of $S_i$ satisfying $T_i \oplus \langle s_i \rangle = S_i$, where $s_i$ is a non-trivial state on an unobservable trajectory $(0, s) \in \mathcal{R}$, without changing the code generated by the realization.

The following theorem, relating state-trimness and non-trimmability is essentially due to Koetter [11, Thm. 9]. It explains why we did not list non-trimmable as a criterion in Theorem 1.

**Theorem 2** $\mathcal{R}$ is non-trimmable if and only if $\mathcal{R}$ is observable and state-trim.

### 3 Trellis Fragments

Trellis fragments play a crucial role in our considerations. They are obtained by cutting a trellis at two state spaces (or at one, in the degenerate case), and thus can be regarded as realizations on cycle-free graphs.

Let $\mathcal{R}$ be a trellis of length $m$ with constraint codes $C_i \subseteq S_i \times A_i \times S_{i+1}$.

Let $(j, k)$ be an interval in $\mathbb{Z}_m$, where $j, k \in \mathbb{Z}_m, j \neq k$. The fragment $\mathcal{R}^{(j, k)}$ is the trellis consisting of all symbol spaces $A_i$ and constraint codes $C_i$ with indices in $(j, k)$, and all state spaces with indices in $(j, k)$. The state spaces $S_i, i \in (j, k)$, are internal state spaces, and the two boundary state spaces $S_j$ and $S_k$ may be regarded as external state spaces.

As for any trellis, the internal behavior $\mathcal{B}^{(j, k)}$ is the set of all trajectories that satisfy all constraints, thus $\mathcal{B}^{(j, k)} = \{(a^{(j, k)}, s^{(j, k)}) \in A^{(j, k)} \times S^{(j, k)} | (s_i, a_i, s_{i+1}) \in C_i \text{ for all } i \in (j, k)\}$, where $A^{(j, k)} \times S^{(j, k)} := \prod_{i \in (j, k)} A_i \times \prod_{i \in (j, k)} S_i$. Such trajectories will be called...
valid paths. The external behavior $C_{(j,k)}$ is the projection $(B_{(j,k)})_{[S_{j} \times A_{(j,k)} \times S_{k}]_{\text{all-zero}}}$. A fragment $R_{(j,k)}$ is thus a normal realization of $C_{(j,k)}$.

It is convenient to extend these definitions to the case where $j = k$, that is, intervals of length $m$ or $0$ “starting at time $j$”. Such fragments arise from cutting the trellis at one edge. We will skip the details and refer to [7, Sec. 5]. All further results are valid and meaningful for these two cases as well.

Consider the dual fragment $(R^{\circ})_{(j,k)}$ with internal and external behavior $(B^{\circ})_{(j,k)}$ and $(C^{\circ})_{(j,k)}$, based on the dual trellis $R^{\circ}$. Due to the definition of the dual constraint codes $C^{\circ}_{j}$ (which implies a sign inversion for the external variable $s_{k}$), normal realization duality yields

$$\{(\hat{s}_{j}, \hat{a}_{(j,k)}, \hat{s}_{k}) \mid (\hat{s}_{j}, \hat{a}_{(j,k)}, -\hat{s}_{k}) \in (C^{\circ})_{(j,k)} \} = (C_{(j,k)})^{\perp}$$

(2)

for any non-empty or non-empty interval $[j, k]$.

The trellis $R$ is called $[j, k]$-controllable if $(C_{(j,k)})_{[S_{j} \times S_{k}]} = S_{j} \times S_{k}$, and $[j, k]$-observable if $(C_{(j,k)})_{[S_{j} \times S_{k}]} = \{(0, 0)\}$. Thus, $R$ is $[j, k]$-controllable if for all state pairs $(s_{j}, s_{k})$ there exists a valid path $(a_{(j,k)}, s_{j}, s_{(j,k)}, s_{k})$, while it is $[j, k]$-observable if there is no nonzero pair $(s_{j}, s_{k})$ for which there is a valid path of the form $(0_{(j,k)}, s_{j}, s_{(j,k)}, s_{k})$.

For $t = 1, \ldots, m$, we will say that a trellis is $t$-controllable (resp. $t$-observable) if it is controllable (resp. observable) on all intervals $[j, j + t]$.

**Theorem 3** Let $R$ be a trellis and $[j, k]$ be any interval.

(a) $R$ is $[j, k]$-controllable iff $R^{\circ}$ is $[j, k]$-observable.

(b) If $R$ is $[k, j]$-controllable, then $B_{(j,k)} = B_{[A_{(j,k)} \times S_{j}]}$ (that is, every valid $[j, k]$-path “lies on a valid trajectory”). The converse is true if $R$ is controllable.

(c) A controllable trellis is $m$-controllable iff it is state-trim.

(d) A controllable trellis is $(m-1)$-controllable iff it is branch-trim.

**Proof** (a) follows from projection/cross-section duality (1) along with (2). For (b) note that $[k, j]$-controllability implies that every valid path in $R_{(j,k)}$ lies on a valid trajectory, as desired. For the converse, project the identity $B_{(j,k)} = B_{[A_{(j,k)} \times S_{j}]}$ on $S_{j} \times A_{(j,k)} \times S_{k}$ to obtain $C_{(j,k)} = B_{[A_{(j,k)} \times S_{j} \times S_{k}]}$. This tells us that every valid path from some state $s_{j}$ to some state $s_{k}$ lies on a valid trajectory, and thus there is also a valid path from $s_{k}$ to $s_{j}$. Hence, $C_{(j,k)}_{[S_{j} \times S_{k}]} \subseteq C_{(k,j)}_{[S_{k} \times S_{j}]}$, and (1), (2) lead to $(C^{\circ})_{(k,j)}_{[S_{k} \times S_{j}]} \subseteq (C^{\circ})_{(j,k)}_{[S_{j} \times S_{k}]}$.

This tells us that for any valid path in $R^{\circ}$ from some $\hat{s}_{k}$ to some $\hat{s}_{j}$ with the all-zero symbol sequence there is a valid path from $\hat{s}_{j}$ to $\hat{s}_{k}$ with the all-zero symbol sequence. Observability of $R^{\circ}$ implies $(C^{\circ})_{(k,j)}_{[S_{k} \times S_{j}]} \subseteq \{(0, 0)\}$. Thus $R^{\circ}$ is $[k, j]$-observable and $R$ is $[k, j]$-controllable. (c) is a special case of (b) because state-trimness is exactly the fact that paths of length 0 lie on valid trajectories. (d) is the case for paths of length 1. $\square$
Our examples illustrate the last two statements: the first trellis in Fig. 1 is not $m$-observable, thus its dual is not $m$-controllable, and hence not state-trim and not branch-trim. The first trellis in Fig. 2 is $m$-observable, but not $(m−1)$-observable, and its dual is state-trim, but not branch-trim.

The following corollary tells us when the dual of a trellis allows a product factorization into elementary trellises. In [12], Koetter and Vardy have shown that a linear trellis allows a product factorization if and only if it is state-trim and branch-trim. We can now characterize when this property is preserved under taking duals.

**Corollary 1** If $R$ is $(m−1)$-observable, then $R^\circ$ is state-trim and branch-trim. If $R$ is observable, but not $(m−1)$-observable, then $R^\circ$ is not branch-trim.

### 4 Reduction Procedures and Irreducibility

Let us formulate the main result of this section. We will sketch parts of the proofs and refer to [7] for details.

The minimum span length $\chi(C)$ of a code $C \subseteq A$ is the minimum length of all possible spans of the nonzero codewords $a \in C$, where a span of $a \neq 0$ is any (possibly circular) interval that covers the support of $a$. For example, $\chi(C) = 3$ for the code $C = \langle 10110, 11001 \rangle \subset F_5^2$.

**Theorem 4** Let $\min\{\chi(C), \chi(C^\perp)\} > t > 1$ and let $R$ be a TPOC trellis of $C$. Then the following are equivalent.

(i) $R$ and $R^\circ$ are $(m−t)$-observable and $(m−t)$-controllable.

(ii) $R$ and $R^\circ$ are $t$-irreducible.

If $R$ is $t$-reducible, then it is strictly $t$-reducible or strictly $(t+1)$-reducible. In either case, $R^\circ$ allows a reduction of the same type.

**Sketch of Proof:** (i) $\Rightarrow$ (ii) is Lemma 1 below. For the converse one observes that by $t$-irreducibility the trellis cannot satisfy the assumptions of Lemma 2. This implies that any unobservable path of length $(m−t)$ must lie on a valid trajectory in $R$. But then the corresponding codeword has a span of length at most $t$, contradicting our assumption. The rest of the theorem follows from Lemma 2.

**Remark 2** Theorem 4 is also true for $t = 1$. In this case the assumption $\min\{\chi(C), \chi(C^\perp)\} > t$ is not needed, and (i) and (ii) are also equivalent to

(iii) $R$ and $R^\circ$ are branch-trim.

It remains to present Lemma 1 and Lemma 2, which form the technical core of our reduction procedure. We start with a sufficient condition for irreducibility.

**Lemma 1** Let $j \neq k$. Suppose that both $R$ and $R^\circ$ are TPOC and $[j,k]$-observable. Then $R$ and $R^\circ$ are $[k,j]$-irreducible.
The main tool for the proof is the fact that a fragment is a realization on a cycle-free graph, and thus minimal if and only if it is trim and proper. Furthermore, all such minimal realizations (with the same graph topology) are isomorphic; see [6]. For the details of this proof we refer to [7, Thm. 6.1].

The other crucial lemma yields a reduction for trellises satisfying a certain technical condition. We restrict ourselves to \([0,m-t)\)-unobservable trellises.

**Lemma 2** Let \(2 \leq t \leq m-1\). Let \(\mathcal{R}\) be a TPOC trellis of length \(m\) that is not \([0,m-t)\)-observable. Thus there exist some \(s_0 \neq 0\), \(s_{m-t} \neq 0\) such that \((s_0, 0, s_{m-t}) \in C(0, m-t)\). Suppose in the fragment \(\mathcal{R}^{[m-t,m-1)}\), there is no valid path from \(s_{m-t}\) to the zero state in \(S_{m-1}\), or there is no valid path from the zero state in \(S_{m-t+1}\) to \(s_0\) in \(\mathcal{R}^{(m-t+1,b)}\). Then \(\mathcal{R}\) has a \(t\)-reduction and a strict \((t+1)\)-reduction. For each of these reductions the dual is a reduction of the same type of the dual trellis \(\mathcal{R}^\circ\).

The proof consists of a reduction procedure. We will sketch the ideas by way of an example and refer to [7, Thm. 6.3] for the details. Consider the trellises in Fig. 2. The second trellis is not branch-trim. Trimming the unused branches in the last section of that trellis and performing the dual process on the first trellis leads to

![Fig. 4](image1.png)  
**Fig. 4** Branch-expanded and branch-trimmed trellises resulting from Fig. 2

Now the second trellis is uncontrollable, and the first one is unobservable. Trimming the first one at time 4 as described in Remark 1 and dually merging the second one leads to

![Fig. 5](image2.png)  
**Fig. 5** State-trimmed and state-merged trellises resulting from Fig. 4

We can continue reducing by using Lemma 2. The first trellis in Fig. 5 has an unobservable path from \(s_0 = 01 \in S_0\) to \(s_3 = 01 \in S_3\), and thus is not \([0,3)\)-observable. It satisfies the assumptions of Lemma 2 in that there is no branch from \(s_3\) to \(00 \in S_4\). As a first step, we expand the state space \(S_4\) to \(S_4^+ = S_4 \oplus \langle \hat{s} \rangle = \{0, 1, \hat{s}, \hat{s} + 1\}\), where \(\hat{s}\) is any new state. We also expand the adjacent constraint codes \(C_3\) and \(C_4\) by adding the branch \((s_3, 0, \hat{s})\) and \((\hat{s}, 0, s_0)\), respectively (and taking the linear hull). This leads to the unobservable trellis shown at the very left of Fig. 6. Next, the assumptions of Lemma 2
guarantee that one can find a subspace $\tilde{S}_4$ of $S_4^+$ such that there is no valid path from $01 \in S_3$ to $\tilde{S}_4$. In this case $\tilde{S}_4 = (1 + \tilde{s})$ is such a subspace. Trimming to $\tilde{S}_4$ leads to the second trellis. Note that this is a non-strict reduction on $[3,0)$ of the first trellis in Fig. 5. Now we can successively trim all unused states and branches. This leads to the last trellis, which is minimal (because conventional and trim and proper). In all these steps the code generated by the trellis does not change.

Fig. 6 Expanding the first trellis in Fig. 5 and trimming the resulting trellis.

We close with the following conjecture, supported by many examples.

**Conjecture** Any nonminimal trellis can be reduced constructively using a finite number of steps as in the reduction procedure from Lemma 2.

**Acknowledgements** I am grateful to G. David Forney, Jr. for an ongoing research collaboration. This abstract is based on [6,7].

**References**

Fuchsian Codes for AWGN Channels

Iván Blanco-Chacón · Dionis Remón · Camilla Hollanti

Received: date / Accepted: date

Abstract We develop a new transmission scheme for additive white Gaussian noisy (AWGN) single-input single-output (SISO) channels without fading based on arithmetic Fuchsian groups. The properly discontinuous character of the action of these groups on the upper half-plane translates into fast decodability.

Keywords Arithmetic Fuchsian groups · AWGN · Coding gain · Decoding · Lattice codes · Quaternion algebras · SISO Channels

Introduction

In the last ten years, a group of people have used number theory as a tool to construct powerful transmission schemes for fast fading MIMO channels. We refer to the use of the structure of orders in number fields and in cyclic division algebras (cf. [3] and [7]).

In the setting of space-time block codes, in [10] it is shown how to use class field theory to derive a lower bound for the minimal determinant of

I. Blanco-Chacón
Aalto University, Department of Mathematics and Systems Analysis, P.O. Box 11100, FI-00076 AALTO, Helsinki, Finland.
Partially supported by Academy of Finland grant 131745.
E-mail: ivnblanco@gmail.com

D. Remón
University of Barcelona, Faculty of Mathematics. Gran Via de les Corts Catalanes 585, 08007 Barcelona (Spain).
Partially supported by MTM2009-07024
E-mail: dremon@gmail.com

C. Hollanti
Aalto University, Department of Mathematics and Systems Analysis, P.O. Box 11100, FI-00076 AALTO, Helsinki, Finland.
Partially supported by Academy of Finland grant 131745.
E-mail: camilla.hollanti@aalto.fi
a space-time block code attached to a cyclic division algebra, and explicit constructions as well as simulations are also carried out. Space-time block codes obtained from cyclic division algebras via conjugations of the left regular representation over a suitable center have non vanishing minimal determinants (NVD property), and this translates into better coding gains. Maximal orders in the context of space-time coding were first considered in [6].

In the present paper we also use some number theoretical ideas to construct a transmission scheme. We assume one transmit (Tx) antenna, one receive (Rx) antenna, and an alphabet consisting of a finite collection of 4-tuples of integers \( \{(x_i, y_i, z_i, t_i)\}_{i=1}^{|C|} \), where \( C \) is the codebook and \( |C| < \infty \) its size. We are interesting in summarizing each 4-tuple into a suitable signal (coding), to send this signal by the antenna, and to reconstruct (decode) the sent 4-tuple. Intuitively, we are sending 4 integers simultaneously by a single antenna. However, our 4-tuple will consist of 3 independent integers, and the fourth will be determined according to these as an additional check-up symbol.

To solve this problem, we will suppose that each 4-tuple determines a matrix belonging to an arithmetic Fuchsian group of the first kind attached to the maximal order (up to conjugation) of an indefinite quaternion algebra which has been fixed at the beginning. This condition translates into the fact that each 4-tuple \((x, y, z, t)\) of our alphabet satisfies an equation of the form \( x^2 - ay^2 - cz^2 + abt^2 = 1 \) for fixed \( a, b \in \mathbb{Z} \). Geometrically, this means that our alphabet is contained in a 4-dimensional hyper-quadric.

Arithmetic Fuchsian groups of the first kind arise in the study of Shimura curves, a rich theory with a large number of theoretical applications in very deep branches of number theory, like in Jacquet-Langlands correspondence, the theory of canonical models or the proof of the Shimura-Taniyama-Weil conjecture (the main argument in Fermat’s Last Theorem). But Shimura curves are also present in the theory of error-correcting codes (cf. [5]).

Fixed a quaternion \( \mathbb{Q} \)-algebra, one can embed the group of units of reduced norm 1 in its maximal order via the left-regular representation into \( M_2(K) \), where \( K \) is a totally real number field, obtaining in this way a discrete group \( \Gamma \).

The crucial properties which yield a remarkably efficient decoding is that the action of \( \Gamma \) on the complex upper half plane (which throughout this paper will be denoted by \( \mathcal{H} \)) is properly discontinuous, and that the third author has recently produced an explicit algorithm to reduce points in \( \mathcal{H} \) to any given fundamental domain, once a presentation of the group has been obtained. In fact, our algorithm does \( O(\log(n)) \) operations, where \( n \) is the number of generating matrices of a given matrix input. We present in this paper a very particular version of this algorithm for a concrete arithmetic Fuchsian group, leaving the general case for forthcoming publications, since it would significantly increase the length of the paper.

The paper is organized as follows: in Section 1 we describe in detail our transmission problem and give some coding theoretical motivations for our code. In Section 2, we review some terminology and facts about orders in quaternion algebras and arithmetic Fuchsian groups which will be used later.
In Section 3 we develop the transmission scheme and explain the algorithm on which it is based. We compute the complexity of this algorithm. In Section 4 we apply the reduction point algorithm to the AWGN channel, and study how to generate the constellation. In Section 5 we plot the performance of our Fuchsian codes with constellation sizes 8 and 16 in terms of signal-to-noise ratio (SNR) versus codeword error rate (CER). Finally, in Section 6 we discuss some topics for further research.

1 Generalities

We are interested in sending 4-tuples of integers \((x, y, z, t)\) subject to the restriction \(x^2 - ay^2 - bz^2 + abt^2 = 1\), where \(a > 0\) and \(b < 0\) and each component can take a finite prescribed number of values. We will send this 4-tuple as the complex signal \(\gamma(\tau)\), where \(\gamma = \left(\frac{x + \sqrt{ay}}{b(z - \sqrt{at})}, \frac{z + \sqrt{at}}{x - \sqrt{ay}}\right)\) and \(\tau\) is an element in the complex upper half-plane \(\mathcal{H}\) to be determined for optimality. We are interested in the case when the matrices belong to a finite subset \(G\) of an arithmetic Fuchsian group \(\Gamma\). In this case, we will refer to the set \(\{\gamma(\tau) | \gamma \in G\}\) as a Fuchsian code. We will take \(\tau\) such that \(\gamma_1(\tau) \neq \gamma_2(\tau)\) if \(\gamma_1 \neq \pm \gamma_2\). To obtain this, it is enough that \(\tau\) is an interior point of a fundamental domain for \(\Gamma\) acting on \(\mathcal{H}\). Hence, in this case, we have as many codewords as 4-tuples.

We will suppose that our channel is affected by additive white Gaussian noise (AWGN), having a channel equation

\[v = u + n,\]

where \(u = \gamma(\tau)\) is the transmitted signal, \(v\) the received signal and \(n \in \mathbb{C}\) is a complex circular Gaussian random variable.

**Definition 1**

Let \(C\) be a code. The data rate in bits per channel use (bpcu) of \(C\) is defined as

\[R = \frac{\log_2(|C|)}{N},\]

where \(N\) the number of channel uses. Notice that in our case \(N = 1\).

The code rate of \(C\) in (real) dimensions per channel use (dpcu) is defined as

\[R_c = \frac{\dim Q(C)}{N},\]

where \(\dim Q(C)\) is the number of independent integer symbols in the code matrix and \(N\) is again the number of channel uses. Notice that in our case the code rate is 3 dpcu.

We will refer to the set of codewords as non uniform Fuchsian (NUF) constellation. It is important to point out that, due to the algebraic dependence of the 4 integers in each of our 4-tuples, we are essentially sending three independent integers at a time by a unique antenna. We will carry out simulations for 8-NUF (3 bpcu) and 16-NUF (4 bpcu) constellations.

In this paper by signal-to-noise ratio we will mean the quotient \(SNR = \frac{E}{N_0}\), where \(E\) is the average energy, i.e., \(E = \frac{1}{|C|} \sum_{k=1}^{|C|} ||z_k||_2^2\), \(\{z_k\}_{k=1}^{|C|} \subset \mathbb{C}\) being the set of the codewords, \(|| \cdot ||_2\) the Euclidean norm, and \(N_0\) the noise variance.
2 Arithmetic Fuchsian groups acting on $\mathcal{H}$

2.1 Quaternion algebras, orders and arithmetic Fuchsian groups

In this section, we survey some facts on the arithmetic of quaternion algebras. We mainly follow the references [1] and [11].

Let $a, b \in \mathbb{Z} \setminus \{0\}$ and let $H = \left( \frac{a,b}{\mathbb{Q}} \right)$ be the quaternion $\mathbb{Q}$-algebra generated by $I$ and $J$ with the standard relations $I^2 = a, J^2 = b, IJ = -JI$. Denote $K = IJ$. The reduced trace and the reduced norm of a quaternion $\omega = x + yI + zJ + tK \in H$ are defined by

\[
\text{Tr}(\omega) = \omega + \overline{\omega} = 2x, \quad N(\omega) = \omega \overline{\omega} = x^2 - ay^2 - bz^2 + abt,
\]

where $\overline{\omega} = x - yI - zJ - tK$ denotes the conjugate of $\omega$. The following map is a monomorphism of $\mathbb{Q}$-algebras

\[
\phi : \left( \frac{a,b}{\mathbb{Q}} \right) \rightarrow M_2(\mathbb{Q}(\sqrt{a}))
\]

\[
x + yI + zJ + tK \mapsto \begin{pmatrix} x + y\sqrt{a} & z + t\sqrt{a} \\ b(z - t\sqrt{a}) & x - y\sqrt{a} \end{pmatrix}.
\]

Notice that for any $\omega \in H$, $N(\omega) = \det(\phi(\omega))$, and $\text{Tr}(\omega) = \text{Tr}(\phi(\omega))$. In the rest of the paper, $H$ will denote a quaternion $\mathbb{Q}$-algebra.

For any place $p$ of $\mathbb{Q}$ (possibly including $p = \infty$), $H_p := H \otimes_{\mathbb{Q}} \mathbb{Q}_p$ is a quaternion $\mathbb{Q}_p$-algebra. If $H_p$ is a division algebra, it is said that $H$ is ramified at $p$. As is well known, $H$ is ramified at a finite even number of places. The discriminant $D_H$ is defined as the product of the primes at which $H$ ramifies. Moreover, two quaternion $\mathbb{Q}$-algebras are isomorphic if and only if they have the same discriminant.

**Definition 2** If $D_H = 1$, $H$ is said to be non-ramified; in this case, it is isomorphic to $M_2(\mathbb{Q})$. If $H$ is ramified at $p = \infty$, it is said to be definite, and indefinite otherwise. An indefinite quaternion algebra is said to be small if $D_H$ is equal to the product of two distinct primes.

An element $\alpha \in H$ is said to be integral if $N(\alpha), \text{Tr}(\alpha) \in \mathbb{Z}$. A $\mathbb{Z}$-lattice $\Lambda$ of $H$ is a finitely generated torsion free $\mathbb{Z}$-module contained in $H$. A $\mathbb{Z}$-ideal of $H$ is a $\mathbb{Z}$-lattice which is a ring. Each order of a quaternion algebra is contained in a maximal order. In an indefinite quaternion algebra, all the maximal orders are conjugate (cf. [11]).

Let us suppose from now on that $H = \left( \frac{a,b}{\mathbb{Q}} \right)$ is indefinite. Given a maximal order $\mathcal{O}_H$, denote by $\mathcal{O}_H^1$ the multiplicative group of elements of $\mathcal{O}_H$ of reduced norm equal to 1, and let $\Gamma_H^1$ be its image under $\phi$.

**Definition 3** Two groups $G_1$ and $G_2$ are said to be commensurable if $G_1 \cap G_2$ has finite index both in $G_1$ and in $G_2$. 
**Definition 4** For $D > 1$, denote by $\Gamma(D, 1)$ the image under $\phi$ of the group of units of reduced norm 1 in a maximal order attached to the quaternion $\mathbb{Q}$-algebra of discriminant $D$. A discrete group $\Gamma \subseteq \text{GL}(2, \mathbb{R})$ is said to be an arithmetic Fuchsian group of the first kind (Fuchsian group from now on) if there exists some quaternion $\mathbb{Q}$-algebra of discriminant $D$ such that $\Gamma$ is commensurable with $\Gamma(D, 1)$. In particular, the groups $\Gamma(D, 1)$ are arithmetic Fuchsian groups of the first kind.

**Remark 1** The notation $\Gamma(D, 1)$ is a particular case of $\Gamma(D, N)$, which stands for the group of units of reduced norm 1 in a special kind of orders which are intersections of two maximal orders, the so called Eichler orders (cf. [1]).

2.2 Fundamental domains

The group $\text{SL}(2, \mathbb{R})$ acts on $\mathbb{H}$ by Möbius transformations and its action factorizes through $\text{PSL}(2, \mathbb{R})$.

**Definition 5** Let $\Gamma$ be an arithmetic Fuchsian group commensurable with $\Gamma(D, 1)$. A fundamental domain for the action of $\Gamma$ on $\mathbb{H}$ is a region $F$ of $\mathbb{H}$ satisfying:

a) For any $z, w \in F$, if there exists $\gamma \in \Gamma$ such that $\gamma(z) = w$, then $z = w$ and $\gamma = \text{Id}$.

b) For any $z \in \mathbb{H}$, there exists $w \in F$ and $\gamma \in \Gamma$ such that $\gamma(z) = w$.

Fundamental domains for several groups $\Gamma(D, 1)$ can be consulted at [1].

3 The point reduction algorithm. Transmission by an absolutely reliable channel

Let us suppose that we have an alphabet consisting of a finite set of 4-tuples of integers, say $A = \{(x_i, y_i, z_i, t_i)\}_{i=1}^N \subset \mathbb{Z}^4$. Let us also suppose that the elements $(x, y, z, t) \in A$ satisfy $x^2 - ay^2 - bz^2 + abt^2 = 1$, the normic equation for $H$. We can think of these 4-tuples as elements of a hyper quadric in $\mathbb{R}^4$.

The way of sending $(x, y, z, t) \in A$ will be by sending the matrix $\gamma = \begin{pmatrix} x + y\sqrt{a} & z + t\sqrt{a} \\ b(z - t\sqrt{a}) & x - y\sqrt{a} \end{pmatrix}$ acting on a point $\tau$ which is interior to a prescribed fundamental domain $F$ and with maximal distance to the boundary of $F$. Notice that we are compressing 4 information symbols, namely, $x, y, z$ and $t$ into $\gamma(\tau)$, we are sending it by an antenna, and hence, we still have three free antennas. If the channel were affected by noise we could use the second and the third antennas to send again $\gamma(\tau)$ (the repetition code), and we would still have the fourth antenna free. But let us assume in this section that our channel is absolutely reliable. The problem is how to decode the received symbol $\gamma(\tau)$ to recover $(x, y, z, t)$, which is equivalent to obtain $\gamma$ out from $\gamma(\tau)$. 
Since $\tau$ is an interior point, and interior points have trivial stabilizers, it suffices to find an algorithm which, given $z \in \mathcal{H}$, returns a representative $w \in \mathcal{F}$ of $z$. This problem is known as point reduction. Since $\pm \gamma$ induce the same map on $\mathcal{H}$, we would recover $\pm \gamma$. Hence, we will assume that we know how to decide if $\gamma$ or $-\gamma$ has been sent. We will prove that this is the case.

### 3.1 Arithmetic Fuchsian groups of signature $(1; e)$

We will develop our algorithm for the so-called arithmetic Fuchsian groups $\Gamma$ of signature $(1; e)$ (for a precise definition cf. Takeuchi [9]). These groups $\Gamma$ are not the image under $\phi$ of a group $\Gamma(D, 1)$, but indeed, the group $\Gamma^2$ generated by products of two elements of $\Gamma$ is. The index $[\Gamma : \Gamma^2]$ is explicitly computed according to a formula by Miyake. This being said, Takeuchi groups cannot be (at first glance) used to encode NUF symbols. The reason why we develop our algorithm for them, is that the general algorithm is more complicated. Nevertheless, we have implemented the algorithm also in the general case and we will plot our SNR/CER graphs with it. The authors are more than satisfied to provide an implementation of our algorithm to any interested reader.

For $g = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma$, if $g$ is not a homothety then denote by $I(g)$ the isometry circle of $g$, namely, the set $\{ z \in \mathcal{H} \mid |cz + d| = 1 \}$. If $g$ is a homothety of factor $\lambda$, then define $I(g) = \{ z \in \mathcal{H} \mid |\lambda z| = 1 \}$. Denote by $\text{Ext}(I(g))$ the exterior of $I(g)$ and by $\text{Int}(I(g))$ the complement of $\text{Ext}(I(g))$. For any $\lambda \in \mathbb{R}$, $\lambda > 0$, denote

$$S(\lambda) = \{ z \in \mathcal{H} \mid \lambda^{-1} \leq |z| \leq \lambda \}.$$

If $h$ is a homothety of factor $\lambda$, then notice that the isometry circles of $h$ and $h^{-1}$ are parallel in the hyperbolic metric. Sometimes (cf. [1]), it is possible to find a system of generators $G$ of $\Gamma$ such that one of them is an hyperbolic homothety $h$ of factor $\lambda$ and a fundamental domain of the form

$$\mathcal{F} = \bigcap_{g \in G \setminus \{h, h^{-1}\}} \text{Ext}(I(g)) \cap S(\lambda).$$

We shall call $S(\lambda)$ the fundamental strip of $\mathcal{F}$. We can construct such a fundamental domain, for instance, when $\Gamma$ is one of the 73 arithmetic Fuchsian groups of signature $(1; e)$, which were classified by Takeuchi in [9]. These arithmetic Fuchsian groups admit a presentation of the form $\Gamma = \langle \alpha, \beta : (\alpha \beta \alpha^{-1} \beta^{-1})^e = \pm 1 \rangle$.

**Proposition 1 (Sijsling, [8])** Let $\Gamma$ be an arithmetic Fuchsian group of signature $(1; e)$ generated by $\alpha$ and $\beta$. Then, after a change of variables, we can suppose that $\alpha$ is a homothety of factor $\lambda$ and $\beta = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$. Furthermore, the hyperbolic rectangle $\mathcal{F} = S(\lambda) \cap \text{Ext}(I(\beta)) \cap \text{Ext}(I(\beta^{-1}))$ is a fundamental domain for $\Gamma$. 
Figure 1 shows a fundamental domain for the group of signature $(1;2)$ labeled \( \Gamma = e_{2d1D6ii} \) in [8]. In this case, 

\[
\alpha = \begin{pmatrix}
\sqrt{3} & 0 \\
0 & \sqrt{2}
\end{pmatrix}
\quad \text{and} \quad
\beta = \begin{pmatrix}
\sqrt{2} & 1 \\
1 & \sqrt{2}
\end{pmatrix}.
\]

Let \( \Gamma \) be an arithmetic Fuchsian group of signature $(1;e)$ generated by \( \alpha, \beta \). Our aim is to develop an algorithm that given \( z \in \mathcal{H} \) returns \( w \in \mathcal{F} \) and \( \gamma \in \Gamma \) such that \( \gamma(z) = w \). This problem is equivalent to this other: given \( g \in \Gamma \), to express \( g \) as a product of powers of \( \alpha \) and \( \beta \). The idea is to multiply \( g \) by the left by a suitable sequence of matrices \( \{g_k\} \), with \( g_k \) a power of \( \alpha \) or \( \beta \) to obtain a product \( g_k \cdots g_k g \), such that \( g_k \cdots g_k g(\tau) \) belongs to the interior of \( \mathcal{F} \), being \( \tau \) as above. In this case, \( g = (g_k g_k \cdots g_k)^{-1} \). Observe that the decomposition of \( g \) as a product of generators is not unique in general.

Let \( z \in \mathcal{H} \) and \( g \in \Gamma \) be such that \( g(z) \notin S(\lambda) \). Define \( N(g) \in \mathbb{Z} \) such that \( \lambda^{-1} \leq |\alpha^{N(g)}(\tau)| \leq \lambda \). We propose the following procedure, whose theoretical justification has been given in [2].

3.2 Complexity

As promised, the properly discontinuous character of the action of the group \( \Gamma \) of signature $(1;e)$ implies fast decodability. Let \( \{(x_k, y_k, z_k, t_k)\}_{k=1}^N \) be the set of integral 4-tuples to be encoded, and \( \{\gamma_k\}_{k=1}^N \) the corresponding set of matrices. Since we have chosen \( \tau \) in the interior of \( \mathcal{F} \), \( |\mathcal{C}| = N \).

**Proposition 2** The total complexity of the point reduction algorithm (counting points, comparisons and additions) is at most \( 10 \log_2 (|\mathcal{C}| + 1) \).

**Proof** In the proof of the correctness of our algorithm, if a matrix is expressible as a product of \( m \) generator matrices, the algorithm does exactly \( m \) steps, each step consisting of at most 2 comparison and 1 matrix product (which accounts for at most 4 products and 4 sums of real numbers). Hence, \( 10m \) operations if the matrix is a product of \( m \) generators.

On the other side, we can write \( |\mathcal{C}| \leq \sum_{k=0}^M 2^k = 2^{M+1} - 1 \) for certain \( M \geq 1 \). Indeed, we can decompose \( |\mathcal{C}| \) as a sum in which the \( k \)-th is at
Algorithm 1 Decomposition into distinguished closed paths

Require: \( g \in \Gamma, \tau \in \text{Int}(\mathcal{F}) \).
Ensure: \( \{n_{\alpha}, n_{\beta}\} \) such that \( \{\tau, g(\tau)\} = n_{\alpha}\{\tau, \alpha(\tau)\} + n_{\beta}\{\tau, \beta(\tau)\} \).

\[
\gamma \leftarrow g, n_{\alpha} \leftarrow 0, n_{\beta} \leftarrow 0;
flag = false.
\]

while flag == false do

if \( \gamma(\tau) \not\in S \) then

\[
n_{\alpha} \leftarrow n_{\alpha} + N(g).
\]

\[
g \leftarrow \alpha N(g), \gamma \leftarrow \gamma - N(g).
\]

else

if \( \gamma(\tau) \in F \) then

flag \leftarrow true.
\]

end if

end if

if \( \gamma(\tau) \in S^+ \) then

\[
n_{\beta} \leftarrow n_{\beta} + 1.
\]

\[
g \leftarrow g^{-1}, \gamma \leftarrow \gamma \beta.
\]

end if

if \( \gamma(\tau) \in S^- \) then

\[
n_{\beta} \leftarrow n_{\beta} - 1.
\]

\[
g \leftarrow \beta g, \gamma \leftarrow \gamma \beta^{-1}.
\]

end if

end while

return \( \{n_{\alpha}, n_{\beta}\} \) such that \( \{\tau, g(\tau)\} = n_{\alpha}\{\tau, \alpha(\tau)\} + n_{\beta}\{\tau, \beta(\tau)\} \).

most the number of matrices which are a product of \( k \) matrices. Hence, the worst case happens when the algorithm needs to decompose a matrix which is product of \( M = \log_2(|C| + 1) \) generators. It performs \( 10M \) operations.

4 The Gaussian channel

Now, we suppose that our channel is affected by an additive white Gaussian noise, which we model as a sequence of independent identically distributed random variables \( \mathcal{CN}(0, \Sigma) \) with \( ||\Sigma|| = N_0 \). In this case, we have \( n \) 4-tuples encoded by matrices \( \{\gamma_1, \ldots, \gamma_n\} \in \Gamma \). Fix a fundamental domain \( \mathcal{F} \) for \( \Gamma \) and let \( \tau \in \mathcal{F} \) be an interior point with maximal distance to the boundary of \( \mathcal{F} \). As in the previous section, if we want to transmit the 4-tuple \( (x_k, y_k, z_k, t_k) \), we actually send the corresponding matrix acting on \( \tau \), i.e., \( \gamma_k(\tau) \). But the receiver will have \( \gamma_k(\tau) + n_k \), where \( n_k \) is a realization of \( \mathcal{CN}(0, \Sigma) \) at the time in which the receiver receives the information. The fact that we have as many matrices as 4-tuples is due to the fact that the point \( \tau \in \mathcal{H} \) on which our matrices act is chosen to be interior to the prescribed fundamental domain \( \mathcal{F} \).

Lemma 1 Let \( \Gamma \) be a Fuchsian group. If \( \gamma_1(\mathcal{F}) = \gamma_2(\mathcal{F}) \) for some \( \gamma_1, \gamma_2 \in \Gamma \), then \( \gamma_1 = \pm \gamma_2 \).
Proof Take an interior point $\tau \in F$. Then, $\gamma_1(\tau) = \gamma_2(w)$, where $w \in F$ is another interior point. Hence, $w = \gamma_2^{-1}\gamma_1(\tau)$. Since interior points cannot be congruent each other modulo $\Gamma$, it follows that $w = \tau$ and $\gamma_1 = \pm\gamma_2$.

Hence, suppose that $w_k$ is small enough to ensure that both $\gamma_k(\tau), \gamma_k(\tau) + n_k \in \gamma_k(F)$. In this case, according with the previous lemma, the reduction algorithm will return the same matrix (up to a sign).

4.1 An alternative Fuchsian group

Here, we explore the Fuchsian code attached to $\Gamma(6, 1)$ (see notations in 2.2.1).

**Theorem 1** (Alsina, Bayer, cf. [1]) For an algebraic integer $\alpha \in \mathbb{Z}[\sqrt{3}]$ we denote by $\alpha'$ its Galois conjugate. Then,

$$ \Gamma(6, 1) = \left\{ \gamma = \frac{1}{2} \begin{pmatrix} \alpha & \beta \\ -\beta' & \alpha' \end{pmatrix} | \alpha, \beta \in \mathbb{Z}[\sqrt{3}], \det(\gamma) = 1, \alpha \equiv \beta \pmod{2} \right\}. $$

This group contains $\Gamma^2$, the group generated by the squares of the elements of the Takeuchi group $\Gamma$ labeled $e_2d_1D_6ii$ in [8]. Furthermore, $\Gamma(6, 1)$ is the image by $\phi$ of the multiplicative group of elements of reduced norm 1 in $\left(\frac{3-1}{3}\right)$. Hence, if $(x, y, z, t)$ is a 4-tuple encoded by an elements of $\Gamma(6, 1)$, it satisfies $x^2 - 3y^2 + z^2 - 3t^2 = 1$.

**Proposition 3** For any $\gamma = \begin{pmatrix} \alpha & \beta \\ -\beta' & \alpha' \end{pmatrix} \in \Gamma(6, 1)$, $\alpha - \beta' \neq 0$.

**Proof** If this were not the case, we would have that $\beta\beta' = 2$, in particular we would have that 2 is a square modulo 3.

Since we have a point reduction algorithm, we have an easy method to decide if the sent matrix is the one that the algorithm returns or we have to change the sign: simply use an Error-Correcting Code to transmit the sign of the sum of the left upper and left lower elements of the matrix.

**Theorem 2** There exists a reduction point algorithm for arithmetic Fuchsian groups of type $\Gamma(D, N)$ of linear complexity in the number of points to be reduced. Thus there exists a Fuchsian code whose decoding complexity is $O(\log g(|C|))$, where $g$ is the minimal number of generators of $\Gamma(D, N)$.

**Proof** The proof uses isometric circles in order to determine which map has to be applied in each steep of the reduction algorithm.

The fundamental domains in these particular cases are given by a number of isometric circles equal twice the number of generators of $\Gamma$. Therefore, in each steep of the reduction algorithm, we perform at most twice the number of generators comparisons and $2 \times 2$ matrix products.
Remark 2 Recall that linearly structured codes are bound to have a worst-case maximum-likelihood (ML) decoding complexity proportional to $|S|^\kappa = |C|$, where $|S|$ is the size of the underlying (e.g., QAM) symbol alphabet, $\kappa$ is the number of independent (QAM) symbols in one codeword, and $|C|$ is the size of the resulting code. Usually linear codes are preferred due to simple decoding methods such as a sphere decoder, but here we have seen that by using nonlinear Fuchsian codes one can actually reduce the decoding complexity from $O(|C|)$ to $O(\log |C|)$.

The constant terms are nevertheless relevant when the code size is small; it can be shown in an analogous way that in the proof of Proposition 2, that the Fuchsian code $\Gamma(6,1)$ has complexity $19 \log_3 (2|C| + 1)$ which is less than $|C|$ for $|C| \geq 90$.

We have to be careful with the choice of $\tau$. If $\gamma(\tau)$ is close to the boundary of $\gamma(F)$, it will be less immune to noise than other point further to the boundary: noise in the direction of minimal distance with this minimal distance as magnitude could eventually bring the point outside the fundamental domain, while this is more difficult to happen to a point of bigger distance to the border. We have adopted the following optimality criterion for choosing $\tau$: denoting by $\partial \gamma_k(F)$ the boundary of $\gamma_k(F)$ and by Int $(\gamma_k(F))$ its interior, let $c_k \in \gamma_k(F)$ be such that $d(c_k, \partial \gamma_k(F)) = \max_{z \in \gamma_k(F)} d(z, \partial \gamma_k(F))$. Then, we have to find $z \in \text{Int}(\mathcal{F})$ minimizing $\sum_k |\gamma_k(z) - c_k|^2$, i.e., the point $\tau$ such that its transforms are simultaneously closest to the corresponding centers on quadratic average. The choice of such an optimal $\tau$ is critical to the performance of the code, as several previous simulations have shown, with non-optimal $\tau$. Notice that different quaternion algebras can yield drastically different performances.

4.2 Generating the constellation

We address now the problem of how to produce the 4-tuples $(x,y,z,t) \in \mathbb{Z}^4$ such that $x^2 - ay^2 - bz^2 + abt^2 = 1$, which will be sent in the form of matrices of $\Gamma(D,1)$ acting on $\tau$ by Möbius transforms. We will restrict ourselves to $a = 3$ and $b = -1$, to be able to derive explicit results, but the same analysis works in general.

As we have said before, only three symbols in each 4-tuple are independent, hence, we would like to parametrize the set of these 4-tuples by an infinite set of 3-tuples $(n,k_1,k_2) \in \mathbb{Z}$. Since the quaternion algebra $\left( \frac{3,-1}{\mathbb{Q}} \right)$ is indefinite, one has that the normic equation $x^2 - 3y^2 + z^2 - 3t^2 = 1$ has infinitely many integer solutions (cf.[1]). It is possible to parametrize all the rational solutions of this normic equation by means of rational functions in three variables, but using this method to produce integer solutions seems a difficult task. We develop instead, an alternative method to produce an infinite set of such solutions. Next, we detail our construction.
First, notice that the ring of integers of the number field \( \mathbb{Q}(\sqrt{3}) \) is \( \mathbb{Z}[\sqrt{3}] \). The multiplicative group of units of this ring is \( \{\pm \varepsilon^n : n \in \mathbb{Z}\} \), where \( \varepsilon \) is a unit of infinite order (called a fundamental unit). This is a very particular version of Dirichlet’s theorem on units, but in our case it is easy to find by inspection such a fundamental unit. We will consider \( \varepsilon = 2 + \sqrt{3} \). Given an element \( \theta = a + \sqrt{3}b \in \mathbb{Q}(\sqrt{3}) \), let us denote by \( \theta' \) its Galois conjugated, i.e., \( a - \sqrt{3}b \).

Given \((n, k_1, k_2)\) a triple of non-negative integers, define \( a_n + \sqrt{3}b_n = \varepsilon^n \). We have that \( a_n^2 - 3b_n^2 = \varepsilon^n (\varepsilon')^n = 1 \). Now, set \( x_{n,k_1} + \sqrt{3}y_{n,k_1} := a_n \varepsilon^{k_1} \) and \( z_{n,k_2} + \sqrt{3}t_{n,k_2} := b_n \varepsilon^{k_2} \). Notice that \( x_{n,k_1}^2 - 3y_{n,k_1}^2 = a_n^2 \) and \( z_{n,k_2}^2 + 3t_{n,k_2}^2 = -3b_n^2 \), hence, the 4-tuple \((x_{n,k_1}, y_{n,k_1}, z_{n,k_2}, t_{n,k_2})\) satisfies

\[
x_{n,k_1}^2 - 3y_{n,k_1}^2 + z_{n,k_2}^2 - 3t_{n,k_2}^2 = a_n^2 - 3b_n^2 = 1.
\]

We will denote by \( \phi(n,k_1,k_2) \) the so constructed 4-tuple. This way, we have parametrized by three variables an infinite subset of points of the hyper quadric \( x^2 - 3y^2 + z^2 - 3t^2 = 1 \).

**Proposition 4** The map \( \phi \) is bijective over its image, which is contained in the set \( \{(x, y, z, t) \in \mathbb{Z}_{>0} : x^2 - 3y^2 = n^2, z^2 - 3t^2 = -3m^2, \text{ for some } n, m \in \mathbb{Z}\} \).

**Proof** Let \((n_1, k_{1,1}, k_{1,2})\) and \((n_2, k_{2,1}, k_{2,2})\) two triples of non-negative integers. Suppose \( n_1 = n_2 = n \). If \( k_{1,1} \neq k_{2,1} \), then \( a_{n_1} \varepsilon^{k_{1,1}} \neq a_{n_2} \varepsilon^{k_{2,1}} \) and \( \phi(n_1,k_{1,1},k_{1,2}) \neq \phi(n_2,k_{2,1},k_{2,2}) \). The case \( k_{2,1} \neq k_{2,2} \) is analogous. Suppose that \( n_1 \neq n_2 \). In this case, \( a_{n_1} \neq a_{n_2} \) or \( b_{n_1} \neq b_{n_2} \). Suppose that \( a_{n_1} \neq a_{n_2} \). In this case, \( a_{n_1} \varepsilon^{k_{1,1}} \neq a_{n_2} \varepsilon^{k_{1,2}} \), since otherwise, \( a_{n_1}^2 = a_{n_2}^2 \), and since \( \varepsilon > 0 \), we would have that \( a_{n_1} = a_{n_2} \). The remaining case is identical.

As an example, we have that \( \phi(1,0,1) = (2,0,3,2), \phi(0,1,1) = (2,1,0,0), \phi(2,0,1) = (7,0,12,8), \) or \( \phi(2,1,1) = (14,7,12,8) \). Notice that the last two values are difficult to obtain merely by inspection.

The explicit parametrization of the whole group of units is a delicate problem. On the contrary to the number field setting, the structure of the group of units of reduced norm 1 in quaternion algebras has not been explicitly described yet. However, there exist some interesting theoretical results, as [4].

### 4.3 Duplicating the size

As a last step in our design, we can duplicate the size of the codebook in the following way: once we have made an optimal choice of matrices of \( \Gamma(D, N) \) and \( \tau \in \mathcal{H} \) having a codebook \( C = \{\gamma_k(\tau)\}_{k=1}^6 \), we can consider the new codebook \( C = \{\pm \gamma_k(\tau)\}_{k=1}^6 \). If a matrix \( \gamma \) corresponds with the 4-tuple \((x, y, z, t)\), and this 4-tuple corresponds to the 3-tuple \((n, k_1, k_2)\) of independent non-negative integers, we can impose that the matrix \(-\gamma\) corresponds to the 3-tuple \((-n, -k_1, -k_2)\). Notice that this is not ambiguous since the original triples are assumed to have non negative entries, and \( \theta > 0 \).
To recover the right 3-tuple from a received signal, first, we check if it belongs to $\mathcal{H}$ or to $-\mathcal{H}$. In the first case, we use the point reduction algorithm to obtain $(x, y, z, t)$ and the parametrization to obtain $(n, k_1, k_2)$. In the second case, we have received (out to outage error) $v = -\gamma_k(\tau) + n$, hence, we apply the point reduction algorithm to $-v$, obtain $(x, y, z, t)$ and $(n, k_1, k_2)$, and we decode it as $(-n, k_1, k_2)$. In the next section we plot the performance of the 8-codes restricted to $\mathcal{H}$ (hence referred to as 4-NUF) and we compare them with the raw sending of 4-QAM symbols, showing in particular that the $\Gamma(10, 1)$-code outperforms 4-QAM and the rest of the codes.

5 Simulation results

![Graph](image1)

**Fig. 2** right: 16-NUF codes, left: 8-NUF codes (brown=$\Gamma$, violet=$\Gamma(6, 1)$, blue=$\Gamma(10, 1)$)

![Graph](image2)

**Fig. 3** right: 4-NUF codes (green=$\Gamma$, brown=$\Gamma(6, 1)$, violet=4-QAM, blue=$\Gamma(10, 1)$), left: 8-NUF constellation for the $\Gamma$-code

6 Further research

As we have seen, the difference in performance for $\Gamma = e2d1D66ii, \Gamma(6, 1), \Gamma(10, 1)$ is remarkable. In forthcoming papers we will compare the performance and complexity of a bigger number of $\Gamma(D, 1)$ and for different code sizes, as well as the explicit parametrization of all the 4-tuples. A further study on our
Fuchsian codes should also include the issue of error correction after point reduction in such a way (but not increasing substantially the complexity). We also remark how important is to choose the matrices with the lowest energy.

6.1 Towards the fading channel

Our scheme is valid only for AWGN channels, since in a fading channel a message $\gamma_k(\tau)$ would reach the receiving end as $h\gamma_k(\tau) + n$, with $h$ a random variable $CN(0, 1)$. One common simplification in the existing literature is to suppose that the receiver has perfect knowledge of the fading coefficient $h$ (via sending pilots in the signal, for instance).

It is well known that we can write $h = re^{i\theta}$ where $r$ is Rayleigh distributed and $\theta$ is uniformly distributed in $[0, 2\pi]$. It is also common to suppose that the noise $n$ originates in the receiver side, hence if we suppose in a first approach that $h = e^{i\theta}$, we can still use our scheme to decode: since $n$ has the same distribution as $e^{-i\theta}n$, i.e., the noise does not get amplified. Hence, it is enough to suppose that $h$ takes values in $\mathbb{R}$ and is Rayleigh distributed.

References


Weight Distribution of Cyclic Codes with Several Non-zeroes

Jinquan Luo

Abstract
In this paper we will study the weight distribution of some cyclic codes which have several non zeroes only.

Index terms: Cyclic code, Weight distribution, Character sum

1 Introduction

These notations are fixed throughout this paper.

- Let \( p \) an odd prime, \( q = p^s \), \( r = q^m \) and \( GF(p^i) \) be the finite field of order \( p^i \). Let \( e \) and \( h \) be two integers and \( eh \mid q - 1 \) and \( \gcd(eh, m) = 1 \). Let \( t \) be an integer coprime to \( e \).

- Let \( g \) be a primitive element of \( GF(r) \) (that is, \( g \) is the generator of the multiplicative group \( GF(r)^* \), \( \alpha = g^h \) and \( \beta = g^{\frac{t^{p^m}}{p^i}} \).

- For \( j \mid i \), let \( \text{Tr}_{p^i/p^j} : GF(p^i) \rightarrow GF(p^j) \) be the trace mapping defined by \( \text{Tr}_{p^i/p^j}(x) = x + x^{p^j} + x^{p^{2j}} + \cdots + x^{p^{j-1}} \).

- Let \( \zeta_p = \exp(2\pi \sqrt{-1}/p) \) be a \( p \)-th root of unity and \( \chi_{p^j}(x) = \zeta_p^{\text{Tr}_{p^i/p^j}(x)} \) be the canonical additive character on \( GF(p^j) \).

The author is with Selmer Center, department of informatics, University of Bergen, N5020, Norway. E-mail address: Jinquan.Luo@ii.uib.no
An \([n, k, d; q]\) linear code is a \(k\)-dimensional \(GF(q)\) linear subspace of \(GF(q)^n\) with minimum Hamming distance \(d\). For an \([n, k, d; q]\) linear code \(C\), let \(A_i\) be the number of codewords in \(C\) with Hamming weight \(i\). The weight distribution \(\{A_0, A_1, \cdots, A_n\}\) is an important research object in coding theory. For instance, code with minimal distance \(d\) can detect up to \(d - 1\) errors and correct up to \([(d - 1)/2]\) errors. Besides, weight distribution can determine the probability of undetected error when the code is used for error detection in an ARQ (automatic retransmission request) system. For convenience, the weight enumerator of \(C\) can be defined by

\[
A_c(x) := A_0 + A_1 x + \cdots + A_n x^n.
\]

In a linear code \(C\), if, for any codeword \((c_0, c_1, \cdots, c_{n-1})\) \(\in C\), the cyclic shifts \((c_i, c_{i+1}, \cdots, c_{i-1})\) for all \(i, 1 \leq i \leq l - 1\) are codewords in \(C\), then \(C\) is called cyclic code. In general, it is hard to obtain the weight distributions of cyclic codes. It is well known that any \(k\)-dimensional \(q\)-ary cyclic code of length \(n\) with \(\gcd(n, q) = 1\) is generated by a polynomial \(g(x) \in GF(q)[x]\) of degree \(n - k\) which is a divisor of \(x^n - 1\), i.e., all the codewords of \(C\) are the vectors of ascending powers’ coefficients of \(a(x)g(x)\) with all \(a(x) \in GF(q)[x]\) and \(\deg a(x) < k\). The reciprocal polynomial \(h(x)\) of \(h^*(x) = (x^n - 1)/g(x)\), i.e., \(h(x) = x^{\deg (h^*(x))}h^*(x^{-1})\) is called the parity check polynomial of \(C\). The zeroes of \(h(x)\) are called the non zeroes of \(C\) (If \(x_0\) is a zero of \(h(x)\), then so is \(x_0^q\). Therefore when we talk about zeroes or non zeroes of a cyclic code, we always refer in particular to a subset of cyclotomic representatives). We say \(C\) is irreducible if \(h(x)\) is irreducible and \(C\) has \(l\) non zeroes only if \(h(x)\) is the product of \(l\) irreducible polynomials. Several classes of cyclic codes and their weight distributions have been investigated for decades, (see Feng and Luo [2], [3], Fitzgerald and Yucas [4], McEliece [6], van der Vlugt [7], van der Vlugt [8], Wolfmann [9], Yuan, Carlet and Ding [10] and the references therein).

Thanks to Delsarte’s Theorem, the weights of codewords in the above \(C\) can be expressed as some kind of combination of additive character sums. Precisely speaking, let \(h_i(x)\) \((0 \leq i \leq l - 1)\) be distinct irreducible polynomials in \(GF(q)[x]\) such that \(g^{-s_i}\) is a zero of \(h_i(x)\) for \(0 \leq i \leq l - 1\), respectively. Let \(h(x) = h_0 \cdots h_{l-1}(x)\) and \(C\) be the cyclic code over \(GF(q)\) with length \(n = (r - 1)/\gcd(s_0, \cdots, s_{l-1})\) and parity-check polynomial \(h(x)\). Then the codewords in \(C\) can be expressed by \(c(a) = (c_0, c_1, \cdots, c_{n-1})\) for \(a = (a_0, \cdots, a_{l-1}) \in GF(q)^l\) where \(c_i = \sum_{j=0}^{l-1} \text{Tr}_{q}(a_j g^{j s_i})\) \((0 \leq i \leq n - 1)\). Therefore the Hamming weight of
the codeword \( c(a) \) is
\[
w_H(c(a)) = \# \{ i \mid 0 \leq i \leq n - 1, c_i \neq 0 \} \\
= n - \# \{ i \mid 0 \leq i \leq n - 1, c_i = 0 \} \\
= n - \frac{1}{q} \sum_{i=0}^{n-1} \sum_{\omega \in GF(q)} \zeta_p \left( \omega^{\frac{i-1}{q}} \text{Tr}_{r/q}(a_j g^{i s_j}) \right) \\
= \left( 1 - \frac{1}{q} \right) n - \frac{1}{q} \sum_{\omega \in GF(q)^*} \sum_{i=0}^{n-1} \chi_r \left( \omega \sum_{j=0}^{l-1} a_j g^{i s_j} \right).
\]

The last equation has an intuitive interpretation: if all the entries of \( c(a) \) are chosen at random, then the expectation of number of nonzero entries in \( c(a) \) is \( \left( 1 - \frac{1}{q} \right) n \). The difference between this expectation and the real weight of \( c(a) \) is formulated as the last term.

In this paper we will study the weight distribution of the cyclic code \( C \) with nonzero entries \( (\alpha \beta)^{-1} \) only for \( 0 \leq i \leq l - 1 \). Note that for the special case \( t = 1 \) (then \( \beta = g^{(r-1)/e} \) and \( l = 2 \), the weight distribution of \( C \) has been determined in Ma et al[5]. Therefore, our result is a generalization of . For abbreviation, denote by
\[
Z(a) = \sum_{\omega \in GF(q)^*} \sum_{i=0}^{n-1} \chi_r \left( \omega \sum_{j=0}^{l-1} a_j (\alpha \beta)^i \right).
\]

Then
\[
w_H(c(a)) = n - \frac{n}{q} - \frac{1}{q} Z(a).
\]

In this way, the weight distribution of cyclic code \( C \) can be derived from the explicit evaluating of \( Z(a) \). Our strategy is the factorization of the multiplicative group generated by \( g^h \) as the cosets of the subgroup generated by \( g^{eh} \) in the summation \( Z(a) \) and applying the weight distribution of Reed Solomon codes.

## 2 Main Result

We follow the notations in Section 1. Let \( G \) be the multiplicative subgroup of \( GF(r)^* \) generated by \( g^h \) and \( H \) be the subgroup of \( G \) generated by \( g^{eh} \). Then we have the following coset factorization
\[ G = \bigcup_{i=0}^{e-1} g^{hi}H. \]

Note that \( GF(q) \) is the multiplicative subgroup of \( GF(r) \) generated by \( g^{(r-1)/(q-1)} \). The following result has been proven in [1] and [5]. We include the proof for the completeness.

**Lemma 1.** For any \( u \in GF(r)^\ast \), there are exactly \( \frac{q-1}{eh} \) pairs \((w, x) \in GF(q)^\ast \times H\) such that \( u = wx \).

**Proof.** Assume \( w = g^{eh} \) and \( x = g^{(r-1)/(q-1)} \) for \( 0 \leq i < eh \) and \( 0 \leq j < q - 1 \). Then the statement is equivalent to saying that for any \( k \), \( 0 \leq k \leq r - 2 \), the equation

\[ eh + \frac{r-1}{q-1} j \equiv k \pmod{r-1} \]  \quad (2)

has \( (q-1)/eh \) solutions for \( i \pmod{(r-1)/eh} \) and \( j \pmod{q-1} \).

Firstly we show that the above equation does have solutions. Since \( eh \) is a divisor of \( q - 1 \) and

\[ \frac{r-1}{q-1} = 1 + q + q^2 + \cdots + q^{m-1} \equiv m \pmod{q-1}, \]

then \( \gcd(eh, (r-1)/(q-1)) = \gcd(eh, m) = 1 \). Therefore the equation have solutions for any \( k \).

Secondly, let \((i_0, j_0)\) be one solution of (2). Then all the solutions of (2) are

\[ (i, j) = (i_0 + \frac{r-1}{q-1} s, j_0 - eh s) \]

for any integer \( s \). If two solutions \((i, j) = (i_0 + \frac{r-1}{q-1} s, j_0 - eh s)\) and \((i', j') = (i_0 + \frac{r-1}{q-1} s', j_0 - eh s')\) give the same factorization of \( u = wx \), then we have

\[ g^{eh(i_0 + \frac{r-1}{q-1} s)} = g^{eh(i_0 + \frac{r-1}{q-1} s')} \]

which is equivalent to

\[ g^{eh \frac{r-1}{q-1} (s-s')} = 1 \]

and then \( r - 1 \mid eh \frac{r-1}{q-1} (s-s') \). Hence we have \( \frac{q-1}{eh} \mid s-s' \). As a consequence the number of different choices of \( s \) which give different \( w \) is \( \frac{q-1}{eh} \) and the result follows. \( \square \)
Note that $\beta = g^{(r-1)/e}$. The Reed-Solomon code $RS(\beta, e, l)$ over $GF(r)$ generated by

$$G_{RS}(\beta, e, l) = \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \beta & \beta^2 & \cdots & \beta^{e-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \beta^{l-1} & \beta^{2(l-1)} & \cdots & \beta^{(e-1)(l-1)}
\end{pmatrix} \tag{3}$$

is an MDS (maximum distance separable) code with parameter $[e, l, e - l + 1; r]$. The weight distribution of $RS(\beta, e, l)$ is as follows.

**Lemma 2.** Let $B_i$ be the number of codewords in $RS(\beta, e, l)$ with weight $i$. Then

$$B_i = \binom{e}{i} (r-1) \sum_{j=0}^{i-e+l-1} (-1)^j \binom{i-1}{j} r^{i-e+l-j-1}.$$

Let $C$ be the cyclic code with only nonzeroes $\alpha \beta^i$ for $0 \leq i \leq l-1$ and $2 \leq l \leq e$. Then the weight of $c(a)$ is obtained by the evaluation of $Z(a)$. Recall that $G$ is the cyclic group generated by $g^h$, $\alpha = g^h$, $\beta = t(r-1)/e$ with $\gcd(t, e) = 1$. Therefore

$$Z(a) = \sum_{\omega \in GF(q)^*} \sum_{x \in G} \chi_r \left( \omega \sum_{j=0}^{l-1} a_j x^{1 + \sum_{k=1}^{i-1} j_k} \right)$$

$$= \sum_{\omega \in GF(q)^*} \sum_{y \in H} \chi_r \left( \omega \sum_{j=0}^{l-1} a_j (g^h y)^{1 + \sum_{k=0}^{i-1} j_k} \right)$$

$$= \sum_{\omega \in GF(q)^*} \sum_{y \in H} \chi_r \left( \sum_{j=0}^{l-1} a_j \beta^{ij} g^h y \right)$$

$$= q^{-1} \sum_{i=0}^{l-1} \sum_{y \in GF(r)^*} \chi_r \left( \sum_{j=0}^{l-1} a_j \beta^{ij} y \right) \tag{4}$$

where the second equation follows from the factorization $G = \bigcup_{i=0}^{e-1} g^{ih} H$, the third equation follows from $y^{(r-1)/en} = 1$ for any $y \in H$ and the last equation follows from Lemma 1.

Denote by $c_i = \sum_{j=0}^{l-1} a_j \beta^{ij}$. Then

$$c'(a) = (c_0, c_1, \cdots, c_{e-1}) = (a_0, a_1, \cdots, a_{e-1}) \cdot G_{RS}(\beta, e, l)$$

5
is a codeword of $\mathcal{RS}(\beta, e, l)$. Note that the inner sum

$$
\sum_{z \in GF(r)^*} \chi_{r}(c_i z) = \begin{cases} 
  r - 1 & \text{if } c_i = 0, \\
  -1 & \text{if } c_i \neq 0.
\end{cases}
$$

Therefore

$$
Z(a) = \frac{q - 1}{eh} ((r - 1) \cdot (e - w_H(c'(a))) - w_H(c'(a)))
$$

and

$$
w_H(c(a)) = \frac{q - 1}{eh} \frac{r - 1}{q} - \frac{1}{q} Z(a) = \frac{(q - 1)q^{n-1}}{eh} w_H(c'(a)).
$$

From the weight distribution of $\mathcal{RS}(\beta, e, l)$, we obtain the weight enumerator of the code $C$ with nonzeros $\alpha\beta^i \ (0 \leq i \leq l - 1 \leq e - 1)$.

**Theorem 1.** The cyclic code $C$ has parameter $[\frac{r - 1}{h}, lm, \frac{q^{n-1}(q-1)}{eh}(e - l + 1); q]$ and its weight enumerator is

$$
A_C(x) = \sum_{i=-l+1}^{e} \binom{e}{i} (r - 1) \sum_{j=0}^{i-e+l-1} (-1)^j \binom{i - 1}{j} r^{i - e + l - j - 1} \cdot x^{\frac{q^{n-1}(q-1)}{eh}i}.
$$

**Proof.** It remains to show that the dimension of $C$ is $lm$. Otherwise there exists some $k, 1 \leq k \leq m - 1$ and $0 \leq i_1, i_2 \leq l - 1 \leq e - 1$ such that

$$
(\alpha\beta^{i_1})^q \equiv (\alpha\beta^{i_2})^q
$$

which is equivalent to

$$g^{(h + it q^{m-1})^q} = g^{(h + it q^{m-1})^q}.
$$

It implies that

$$
(h + it q^{m-1})^q \equiv h + it q^{m-1} \pmod{q^m - 1}.
$$

Reducing to mod $(q^m - 1)/e$, we obtain

$$
h q^k \equiv h \pmod{(q^m - 1)/e}
$$

and equivalently,

$$
\frac{q^{m-1}}{eh} | q^k - 1
$$

which contradicts to $\frac{q^{m-1}}{eh} \geq \frac{q^{m-1}}{q - 1}$ and $q - 1 \leq q^k - 1 \leq q^{m-1} - 1 < \frac{q^{m-1}}{q - 1}$. □
Remark. (1). When $l = 1$, then the code $C$ is the Simplex code which has only one nonzero weight.

(2). When $l = 2$ and $t = 1$, the code $C$ has been studied in Ma et al [5]. This special code has two nonzero weights.

(3). In general, the code $C$ has $l$ nonzero weights: $\frac{q^{m-1}(q-1)}{e}i$ for $e-l+1 \leq i \leq e$.

Example 1. When $q = 7$, $m = 2$, $e = l = 3$ and $h = 1$, the code $C$ has parameters [48, 6, 14; 7]. Using Magma, we can calculate the weight enumerator of $C$

$$A_C(x) = 1 + 144x^{14} + 6912x^{28} + 117649x^{42}$$

which coincides with Theorem 1.

3 Acknowledgements

The author thanks the anonymous referees for their helpful comments which improve the presentation of the paper.

The work of J.Luo is supported by Norwegian Research Council (No. 191104/v30), NSF of China under grant 60903036, NSF of Jiangsu Province under grant 2009182 and the open research fund of National Mobile Communications Research Laboratory, Southeast University (No. 2010D12).

References


On transform–domain decoding of Gabidulin codes

Wenhui Li · Vladimir Sidorenko · Di Chen

Received: December, 20, 2012

Abstract Gabidulin codes are the rank metric analogues of Reed–Solomon codes and found many applications including network coding. Interleaving or the direct sum of Gabidulin codes allows both decreasing the redundancy and increasing the error correcting capability for network coding. In this paper, for Gabidulin codes we propose a transform–domain algorithm correcting both errors and erasures. We show how to generalize this algorithm for interleaved Gabidulin codes. The transform–domain approach allows to simplify derivations and proofs and also simplifies finding the error vector after solving the key equation.

1 Introduction

Gabidulin codes [1], [2], [3] are closely related to Reed–Solomon codes and have maximum possible distance in rank metric. These codes have many applications and recently they have attracted significant attention since they can provide a near-optimal solution to the error control problem in network coding [4], [5]. It was shown in [6] that interleaving or the direct sum of \( L \) Gabidulin codes allows both decreasing the redundancy and increasing the error correcting radius nearly twice. So, it is important to have efficient decoding of Gabidulin codes and their interleaving.

There are known a number of error correcting algorithms for Gabidulin codes similar to Reed–Solomon codes: “a standard” method based on Berlekamp–Massey like [7], [8] or on Sugiyama et al. like [1] approaches, in contrast to...
Welch–Berlekamp like [9] algorithm or Gao like algorithm [10]. We will consider the standard approach, when first a key equation should be solved to find “positions” of errors, and second the error vector can be computed. In contrast to Reed–Solomon codes, where the second step is relatively simple, for Gabidulin codes, the second step is very complicated for both understanding and for computations. In [11], Silva and Kschischang suggested an elegant solution for the second decoding step using a transform–domain approach. This approach allows to simplify derivation and proofs of the second decoding step and also allows to decrease the decoding complexity.

In contrast to Hamming metric, it is not obvious how to define erasures in rank metric. The definition of erasures for rank metric was proposed in [5] and [12]. This definition arises from network coding applications and generalizes the previous definition, where it was assumed that a number of rows and columns in the code matrix are erased. The most decoding algorithms were obtained for correction errors only without erasures. However, in [5] and [12] the standard decoding algorithms were generalized for the case of errors and erasures. These algorithms work in the “time–domain” where the second decoding step (finding the error vector after solving the key equation) is complicated. Error and erasure correction algorithms for the transform–domain were not known.

In this paper, for a Gabidulin code we propose a transform–domain algorithm correcting both errors and erasures. We also show how to generalize this algorithm for interleaved Gabidulin codes.

2 Gabidulin Codes

We consider a finite field \( F = \mathbb{F}_{q^m} \), where \( q \) is power of a prime, with the Frobenius automorphism \( \theta(a) = a^q \) for \( a \in \mathbb{F}_{q^m} \). For an integer \( i \) we define \( \theta^i = \theta^{i \mod m}, \theta^0(a) = a, \) and \( \theta^i(a) = \theta(\theta^{i-1}(a)) \).

Given a basis of the field \( \mathbb{F}_{q^m} \) over the subfield \( \mathbb{F}_q \), we represent every element of \( \mathbb{F}_{q^m} \) as a column vector of length \( m \) over \( \mathbb{F}_q \). In this way every row vector \( v \) over \( \mathbb{F}_{q^m} \) will be written as a matrix \( V \) over \( \mathbb{F}_q \). The rank norm (weight) \( \text{rank}_q v \) of a vector \( v \) is defined to be \( \text{rank}_q V \). The rank distance between two vectors \( v, w \in \mathbb{F}_{q^m}^n \) is defined as \( \text{rank}_q (V - W) \).

For \( n \leq m \), let us fix the vector

\[
  h = (h_1, h_2, \ldots, h_n).
\]

with components \( h_i \in \mathbb{F}_{q^m} \) linearly independent over \( \mathbb{F}_q \), and define the following \( n \times n \) transform matrix \( \Phi \) over \( \mathbb{F}_{q^m} \)

\[
  \Phi = \begin{pmatrix}
    h_1 & h_2 & \ldots & h_n \\
    \theta(h_1) & \theta(h_2) & \ldots & \theta(h_n) \\
    \vdots & \vdots & \ddots & \vdots \\
    \theta^{n-1}(h_1) & \theta^{n-1}(h_2) & \ldots & \theta^{n-1}(h_n)
  \end{pmatrix}.
\]

The Moore matrix \( \Phi \) is non-singular [13] and has the inverse matrix \( \Phi^{-1} \).
Definition 1 A Gabidulin code $G$ is a linear $(n,k)$ code of length $n$ and dimension $k$ over the field $\mathbb{F}_{q^m}$, $n \leq m$, with parity check matrix $H$, i.e.,

$$G(q^m; n,k) = \{ c \in \mathbb{F}_{q^m}^n : cH^T = 0 \},$$

where the $(n - k) \times n$ parity check matrix $H$ consists of the first $n - k$ rows of the matrix $\Phi$.

The distance of a Gabidulin $(n,k)$ code in rank metric is $d = n - k + 1$ and it reaches the Singleton type upper bound.

3 Skew polynomials

To work with Gabidulin codes it will be convenient to use the skew polynomial ring of automorphism type [14] which we now define. Given a field $\mathbb{F}$ and an automorphism $\theta$ of $\mathbb{F}$, one defines a ring structure on the set

$$\mathbb{F}[x; \theta] = \{ a(x) = a_n x^n + \cdots + a_1 x + a_0 \mid a_i \in \mathbb{F} \text{ and } n \in \mathbb{N} \}.$$

This is the set of formal polynomials where the coefficients are written on the left of the variable $x$. The addition in $\mathbb{F}[x; \theta]$ is defined to be the usual addition of polynomials and the multiplication is defined by the basic rule $xa = \theta(a)x$ and extended to all elements of $\mathbb{F}[x; \theta]$ by associativity and distributivity.

Given a skew polynomial $a(x)$ of degree $n$, we define a reciprocal skew polynomial $a^\ast(x)$ having coefficients $a^\ast_i = \theta^i(a_{n-i})$ for $i = 0, \ldots, d$.

In our case of the finite field and the Frobenius automorphism $\theta$, there is a ring isomorphism between the ring $\mathbb{F}[x; \theta]$ of skew (or twisted) polynomials and the ring of linearized polynomials. The last term is used more often in coding theory literature. Given a skew polynomial $a(x) \in \mathbb{F}[x; \theta]$, we denote the corresponding linearized $q$-polynomial by $a_q(x)$, where

$$a_q(x) = \sum_{j=0}^{n} a_j \theta^j(x) = a_n x^{q^n} + \cdots + a_1 x^{q^1} + x,$$

and $n$ is called the $q$-degree of $a_q(x)$ if $a_n \neq 0$.

All polynomials in the paper belong to the ring $\mathbb{F}[x; \theta]$, except for their linearized polynomial equivalents denoted by the lower index ($q$). Given $\theta$, by $M(N)$ we denote the complexity of the multiplication of two skew polynomials of maximum degree $N$.

Later we show that to solve the key equation we need to solve the following problem.

Problem 1 (Skew shift-register synthesis) Given a field $\mathbb{F}$, an automorphism $\theta$ of the field, and $L$ sequences $s^{(1)}, s^{(2)}, \ldots, s^{(L)}$ over $\mathbb{F}$ with lengths $N^{(1)}, N^{(2)}, \ldots, N^{(L)}$, respectively, find the smallest nonnegative integer $\lambda$ for
which there is a vector of coefficients $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_\lambda)$ over $F$ such that for $\ell = 1, 2, \ldots, L$

$$s_n^{(\ell)} = -\sum_{i=1}^\lambda \sigma_i \theta^i \left(s_{n-1}^{(\ell)}\right)$$ for $n = \lambda + 1, \ldots, N^{(\ell)}$. \hspace{1cm} (3)

Moreover, find a connection vector $\sigma$ which fulfills (3).

Given a connection vector $\sigma = (\sigma_1, \ldots, \sigma_\lambda)$ over $F$, we define the corresponding connection polynomial $\sigma(x) \in F[x;\theta]$ as follows

$$\sigma(x) \triangleq 1 + \sigma_1 x^1 + \cdots + \sigma_\lambda x^\lambda,$$

where $\sigma_0 = 1$ and $\deg \sigma(x) \leq \lambda$.

An effective Berlekamp–Massey type algorithm to solve Problem 1 was suggested in [6]. This algorithm has time complexity $O(LN^2)$ operations in $F$, where $N$ is the length of a longest sequence. A fast algorithm for solving Problem 1 was suggested in [15]. The asymptotic time complexity of this algorithm is $O(M(N) \log N)$, for a fixed number of sequences.

4 Decoding a single code with errors and erasures

Assume a codeword $c = (c_1, \ldots, c_n) \in F_q^m$ of the code $G^m(n, k)$ was transmitted and a word $r = (r_1, \ldots, r_n) \in F_q^m$ was received. It means that the error word is $e = (e_1, \ldots, e_n) \in F_q^m$, where $e = r - c$.

We assume that the decoder has some side information about the error vector $e$. More precisely, we assume that $e$ can be represented as follows

$$e = a_F B_F + a_R B_R + a_C B_C. \hspace{1cm} (4)$$

Here the error vector $e$ has unknown $q$-rank $t$, where $t = \epsilon + \delta + \mu$, $\delta$ and $\mu$ are known,

$$a_F \in F_q^{\epsilon \times m}, B_F \in F_q^{\epsilon \times n}, \text{rank}_q a_F = \text{rank}_q B_F = \epsilon,$$

where $\epsilon$, $a_F$ and $B_F$ are unknown, and vector $e_F = a_F B_F$ is called the full error of rank $\epsilon$,

$$a_R \in F_q^{\delta \times m}, B_R \in F_q^{\delta \times n}, \text{rank}_q a_R = \text{rank}_q B_R = \delta,$$

where $a_R$ is known, $B_R$ is unknown, and vector $e_R = a_R B_R$ is called the row erasure of known rank $\delta$, since $\delta$ erased rows at known positions can be represented in this form,

$$a_C \in F_q^{\mu \times m}, B_C \in F_q^{\mu \times n}, \text{rank}_q a_C = \text{rank}_q B_C = \mu,$$

where $a_C$ is unknown, $B_C$ is known, and vector $e_C = a_C B_C$ is called the column erasure of known rank $\mu$, since $\mu$ erased columns at known positions can be represented in this form. As a result we can rewrite (4) as

$$e = e_F + e_R + e_C, \hspace{1cm} (5)$$

or also as

$$e = aB = (a_C, a_F, a_R) (B_C, B_F, B_R)^T. \hspace{1cm} (6)$$
4.1 Modified key equation

In [5] and [12] the following definitions were introduced to get a modified key equation incorporating errors and erasures. Let us introduce skew polynomials

$$\sigma^F(x), \sigma^R(x), \text{and} \quad \sigma^{FR}(x) = \sigma^F(x) \sigma^R(x)$$

such that correspondent $q$-polynomials $\sigma_{FR(q)}(x)$, and $\sigma_{R(q)}(x)$ are linearized polynomials of smallest $q$-degree, satisfying

$$\sigma_{R(q)}(a_{R,i}) = 0, \quad i = 1, \ldots, \delta,$$
$$\sigma_{FR(q)}(a_{F,i}) = 0, \quad i = 1, \ldots, \epsilon.$$

Using known vector $a_R$ of row erasures, the polynomial $\sigma_{R(q)}(x)$ of minimum $q$-degree $\delta$ can be found as (see [5] for the procedure minpoly(.))

$$\sigma_{R(q)}(x) = \text{minpoly}(a_R).$$

Define

$$f = (f_1, \ldots, f_\mu) = Bch^T,$$

and $\lambda_C(x)$ to be a linearized polynomial of minimal $q$-degree $\mu$ satisfying

$$\lambda_{C(q)}(f_i) = 0, \quad i = 1, \ldots, \mu, \text{ hence } \lambda_{C(q)}(x) = \text{minpoly}(f).$$

We compute the syndrome vector $s$ as usual

$$s = (s_1, \ldots, s_{d-1}) = rH^T = eH^T,$$

and introduce the syndrome polynomial $s(x)$

$$s(x) = \sum_{i=1}^{n-k} s_ix^{i-1}. \hspace{1cm} (11)$$

Finally, as in [5], we define the modified syndrome polynomial $s_{RC}(x)$, which incorporates known information about row and column erasures

$$s_{RC}(x) = \sigma_R(x)s(x)\lambda_C(x)$$

and get the modified key equation.

**Theorem 1 (Key equation for Gabidulin codes)** The following equation holds

$$\sigma^F(x)s_{RC}(x) \equiv \omega(x) \mod x^{n-k}, \hspace{1cm} (13)$$

where the error evaluator polynomial $\omega(x)$ is defined by the first $t$ components of the modified syndrome $s_{RC}(x)$ and $\deg \omega(x) < t$.

Here and later $\mod$ means the right modulo operation in $\mathbb{F}[x; \theta]$. The key equation (13) can be equivalently written as

**Corollary 1** For $j = \mu + \delta + \epsilon + 1, \ldots, n - k$ the following equations hold

$$\sum_{i=0}^{\epsilon} \sigma_{F,i} \theta^i(s_{RC,j-i}) = 0. \hspace{1cm} (14)$$
4.2 Decoding in the transform domain

Let us introduce the transformed error vector and polynomial

\[ \tilde{e} = e \Phi^T, \quad \tilde{e}(x) = \sum_{i=1}^{n} \tilde{e}_i x^{i-1}. \]  \hspace{1cm} (15)

To find the transformed error vector \( \tilde{e} \) and then the error vector \( e \), we propose the following key equation for \( \tilde{e} \).

**Theorem 2** The transformed error polynomial \( \tilde{e}(x) \) satisfies the following equation

\[ \sigma_{FR}(x) \tilde{e}(x) \lambda_C(x) \equiv \omega(x) \mod x^n, \]  \hspace{1cm} (16)

where the polynomial \( \omega(x) \) is defined by (13).

**Proof** Given the Gabidulin \((n,k)\) code and an error vector \( e \) with fixed decomposition (4), then from Theorem 1, we can compute the error evaluator polynomial \( \omega(x) \) as follows

\[ \omega(x) \equiv \sigma_F(x) s_{RC}(x) \mod x^{n-k}. \]

From (12) and (7)

\[ \sigma_F(x) s_{RC}(x) = \sigma_F(x) \sigma_R(x) s(x) \lambda_C(x) = \sigma_{FR}(x) s(x) \lambda_C(x), \]

and we obtain

\[ \sigma_{FD}(x) s(x) \lambda_C(x) \equiv \omega(x) \mod x^{n-k}. \]  \hspace{1cm} (17)

Consider another Gabidulin \((n,0)\) code \( G' \) with \( k' = 0 \) and the same as before error vector \( e \) with the same decomposition (4). In this case the polynomials \( \sigma_{FD}(x) \) and \( \lambda_C(x) \) stay unchanged. The parity check matrix \( H' \) of this code is \( H' = \Phi \), and from (10) the syndrome vector \( s' \) of length \( n \) coincides with the transformed error vector \( \tilde{e} \). Since the first \( n - k \) components of syndromes \( s \) and \( s' = \tilde{e} \) coincide, the polynomial \( \omega'(x) \) for the new code coincides with \( \omega(x) \) by Theorem 1. As a result, the theorem statement follows from (17) and Theorem 1 applied to the new code \( G' \).

We are ready to describe our decoding algorithm shown by Algorithm 1.

In Lines 3-6 we compute erasure polynomials \( \sigma_R(q)(x), \lambda_C(q)(x) \) and the modified syndrome \( s_{RC}(x) \) using (8), (9) and (12).

In Line 7 we find \( \sigma_F(x) \) by solving the key equation (14). To do this we observe that according to Corollary 1 solving the key equation is equivalent to solving Problem 1 with parameters \( L = 1, \ N^{(1)} = n-k, \ s^{(1)} = s \). Problem 1 can be solved using algorithms in [16], [6], [15], and we get the number of full errors \( \epsilon = \lambda \), and the full error span polynomial \( \sigma_F(x) = \sigma(x) \). In addition, these algorithms indicate when the found polynomial \( \sigma(x) \) is not unique. In this case we declare decoding failure.

In Lines 8–9 we compute the error evaluator polynomial \( \omega(x) = \sigma_F(x) s_{RC}(x) \mod x^{n-k} \) using (13) from Theorem 1 and the polynomial \( \sigma_{FR}(x) \) using (7).
Algorithm 1: Decoding of a single Gabidulin code

1. **input:** Received word \( r \in \mathbb{F}_q^n \), vector \( a_R \) of row erasures, matrix \( B_C \) of column erasures

2. **begin**
3. Row erasure polynomial: \( \sigma_{R(i)}(x) = \text{minpoly}(a_R) \)
4. Column erasure polynomial: \( f = B_C h^T \), \( \lambda_{C(q)}(x) = \text{minpoly}(f) \)
5. Syndrome: \( s = r H^T \)
6. Modified syndrome: \( s_{RC}(x) = \sigma_R(x) s(x) \bar{\lambda}_C(x) \)
7. Find \( \sigma_F(x) \) by solving the key equation (14) using the Berlekamp–Massey type algorithm in [6]; in case of non single solution output decoding failure
8. The error evaluator polynomial \( \omega(x) = \sigma_F(x) s_{RC}(x) \mod x^{n-k} \)
9. \( \sigma_{FR}(x) = \sigma_F(x) \sigma_R(x) \)
10. \( s_C(x) = \sigma_{FR}(x) \mod x^{n-1} \)
11. The transformed error word \( \tilde{e}(x) = s_C(x) / \bar{\lambda}_C(x) \mod x^{n-1} \)
12. The error word \( e = \tilde{e} \left( \Phi^{-1} \right)^T \)
13. **end**

14. **output:** The codeword \( c = r - e \) or decoding failure

In Lines 10–11 we compute the transformed error word using Theorem 2 as follows. First we compute auxiliary polynomial \( s_C(x) = \sigma_{FR}(x) \mod x^{n-1} \) by left power series expansion of skew polynomials. We write \( a(x) \mod b(x) = c(x) \) if \( b(x) = a(x) c(x) \), and \( c(x) \mod x^{n-1} \) means that from the power series \( c(x) \) we only take items \( c_i x^i \) for \( i = 0, \ldots, n-1 \). Finally, we compute the transformed error word \( \tilde{e}(x) = s_C(x) / \bar{\lambda}_C(x) \mod x^{n-1} \) by right power series expansion.

In Line 12 we compute the error word \( e = \tilde{e} \left( \Phi^{-1} \right)^T \) by the inverse transform.

From the above derivations similar to [5] and [12] we obtain the following theorem.

**Theorem 3** Algorithm 1 corrects \( \epsilon \) full errors, \( \delta \) row erasures and \( \mu \) column erasures as long as

\[ 2 \epsilon + \delta + \mu \leq n - k = d - 1. \]  \hspace{1cm} (18)

Time complexity of Algorithm 1 is \( \mathcal{O}(n^2) \) operations in \( \mathbb{F}_q^m \) for \( m, n, d \) of the same order, see also [5] and [17] for details. We would like to emphasize that after the key equation is solved in Line 7 and the error span polynomial \( \sigma_F(x) \) is found, we need only two polynomial multiplications and two power series expansions, which is equivalent to two divisions of skew polynomials to find the error word in the transform–domain. Then the error vector can be computed by the low complexity inverse transform [11]. Recall that in previous algorithms [5] and [12] instead of these steps one should do the following more complicated steps:

- One polynomial multiplication to find \( \sigma_{FR}(x) \),
- Solve a system of linear equations (41) in [5] to find \( \beta = (\beta_1, \ldots, \beta_\mu) \),
- Compute \( \sigma_{C(q)}(x) = \text{minpoly}(\beta) \),
- Compute \( \sigma(x) = \sigma_C(x) \sigma_F(x) \sigma_R(x) \),
Find a basis for the root space of \( \sigma(q)(x) \),
- Solve a system of linear equations (36) in [5] to find error locators,
- Compute error locations,
- Compute the error word.

As a result, the algorithms in [5] and [12] are more difficult for understanding, despite they have the same order of time complexity \( O(n^2) \).

5 Interleaved Gabidulin Codes

Let us consider interleaving of \( L \) identical Gabidulin codes \( G(q^m; n, k) \) to simplify notations. Generalization for the case of different \((n^{(\ell)}, k^{(\ell)})\) Gabidulin codes over \( \mathbb{F}_{q^m} \) is straightforward and will be mentioned later. Let us agree that the index \( \ell \) runs always from 1 to \( L \), i.e., \( \ell = 1, \ldots, L \). Denote by \( c = (c^{(1)} \ldots c^{(L)}) \in \mathbb{F}^{Ln}_{q^m} \) the concatenation of \( L \) vectors \( c^{(\ell)} \).

**Definition 2** The \( L \)-interleaved Gabidulin code is

\[
IG(q^m, L; n, k) \triangleq \left\{ (c^{(1)} \ldots c^{(L)}) : c^{(\ell)} \in G(q^m; n, k) \right\}.
\]

The interleaved code is defined over \( \mathbb{F}_{q^m} \) and has length \( Ln \) and the distance in rank metric \( d = n - k + 1 \), which reaches the Singleton upper bound if \( m = n \). Recall that Loidreau and Overbeck [18], [19], suggested another construction of an interleaved code, which is over \( \mathbb{F}_{q^{Lm}} \) and has length \( n \).

Assume a codeword \( c \in IG(q^m, L; n, k) \) was transmitted and a word \( r = (r^{(1)} \ldots r^{(L)}) \in \mathbb{F}^{Ln}_{q^m} \) was received. It means that the error word is \( e = (e^{(1)} \ldots e^{(L)}) \in \mathbb{F}^{Ln}_{q^m} \), where \( e = r - c \). Denote \( \text{rank}_q e = t \), then we can represent the error word as

\[
e = aB, \quad \text{where} \ a \in \mathbb{F}^t_{q^m}, \ B \in \mathbb{F}^{t \times Ln}_{q},
\]

where for every component code we have the following error vector

\[
e^{(\ell)} = a_F B^{(\ell)}_F + a_R B^{(\ell)}_R + a_C B^{(\ell)}_C,
\]

where \( a_R \) and \( B^{(\ell)}_C \) are known, and hence \( \delta = \text{rank}_q a_R \) and \( \mu^{(\ell)} = \text{rank}_q B^{(\ell)}_C \) are also known.

We define error and erasure span polynomials and syndrome polynomials in a way similar to a single code. Pay attention that the vector \( a \) is common for all interleaved codes, and this allows us to get more equations for the common error span polynomial \( \sigma_F(x) \). In a way similar to a single code, we obtain a key equation for the interleaved Gabidulin codes as follows.

**Theorem 4 (Key equation for interleaved Gabidulin codes)** The following equation holds for \( \ell = 1, \ldots, L \)

\[
\sigma_F(x) s^{(\ell)}_{RC}(x) \equiv \omega^{(\ell)}(x) \mod x^{n-k},
\]

where the error evaluator polynomial \( \omega^{(\ell)}(x) \) is defined by the first \( t \) components of the modified syndrome \( s^{(\ell)}_{RC}(x) \) and \( \deg \omega^{(\ell)}(x) < t \).
Decoding algorithm for interleaved Gabidulin codes is given by Algorithm 2, which is similar to the one for a single code. In the new algorithm, after the key equation (20) is solved and the error span polynomial $\sigma_F(x)$ is found, the error words $e^{(\ell)}$ can be computed separately for every interleaved code as in Algorithm 1 applying Theorem 2 for every component code:

$$\sigma_{FR}(x)e^{(\ell)}(x) \equiv \omega^{(\ell)}(x) \mod x^n,$$

where the polynomials $\omega^{(\ell)}(x)$ are defined by (20).

**Algorithm 2: Decoding of interleaved Gabidulin codes**

1. **input:** Received word $r = (r^{(1)}, \ldots, r^{(L)}) \in \mathbb{F}_q^{Ln}$, $a_R$, $B_\ell^{(q)}$, $\ell = 1, \ldots, L$
2. **begin**
3. \hspace{1em} Row erasure polynomial: $\sigma_{R(q)}(x) = \text{minpoly}(a_R)$
4. \hspace{1em} for $\ell = 1, \ldots, L$ do
5. \hspace{2em} Column erasure polynomials: $f^{(\ell)} = B_\ell^{(q)}h^T$, $\lambda^{(\ell)}(x) = \text{minpoly}(f^{(\ell)})$
6. \hspace{2em} Syndromes: $s^{(\ell)} = r^{(\ell)}H^T$
7. \hspace{2em} Modified syndromes: $s^{(\ell)}_{RC}(x) = \sigma_{R(x)}s^{(\ell)}(x)\lambda^{(\ell)}_C(x)$
8. \hspace{1em} Find $\sigma_F(x)$ by solving the key equation (20) using the Berlekamp–Massey type algorithm in [6]; in case of non single solution output decoding failure
9. \hspace{1em} for $\ell = 1, \ldots, L$ do
10. \hspace{2em} The error evaluator polynomial: $\omega^{(\ell)}(x) = \sigma_F(x)s^{(\ell)}_{RC}(x) \mod x^{n-k}$
11. \hspace{2em} $\sigma_{FR}(x) = \sigma_F(x)\sigma_R(x)$
12. \hspace{2em} $s^{(\ell)}_C(x) = (\sigma_{FR}(x)\omega^{(\ell)}(x))^{n-1}_0$
13. \hspace{2em} The transformed error word: $\tilde{e}^{(\ell)}(x) = \tilde{s}^{(\ell)}(x)\lambda^{(\ell)}_C(x)^{n-1}_0$
14. \hspace{2em} The error word: $e^{(\ell)} = \tilde{e}^{(\ell)}(\Phi^{-1})^T$
15. \hspace{1em} $e = (e^{(1)}, \ldots, e^{(L)})$
16. **end**
17. **output:** The codeword $c = r - e$ or decoding failure

Using [6] we can summarize our results as follows.

**Theorem 5** The fraction $P_f(t)$ of full error vectors of rank $e_F = \epsilon$ uncorrectable by Algorithm 2 in presence of $\delta$ row erasures and $\mu_1, \ldots, \mu_L$ column erasures is upper bounded by

$$P_f(t) \leq 3.5q^{-m((L+1)(t_{max}-t)+1)} < \frac{4}{q^m}$$

if

$$L \leq \epsilon \leq t_{max} = \frac{L}{L+1}(d-1)$$

and

$$P_f(t) = 0 \text{ for } t < d_{min}/2,$$
where

\[
\bar{d} = \frac{1}{L} \sum_{\ell=1}^{L} d^{(\ell)} - \delta - \mu^{(\ell)}, \quad d_{\text{min}} = \min_{\ell} \{d^{(\ell)} - \delta - \mu^{(\ell)}\}
\]

are the average and minimum code distances respectively after erasings in interleaved \((n^{(\ell)}, k^{(\ell)})\) Gabidulin codes having distances \(d^{(\ell)} = n^{(\ell)} - k^{(\ell)} + 1\).

This means that errors of rank weight less than \(d_{\text{min}}/2\) are always corrected by the decoding algorithm, and all other errors of weight up to the decoding radius \(t_{\text{max}}\) are corrected with very high probability. For the fixed order of interleaving \(L\), the asymptotic complexity of Algorithm 2 is the same as the one of Algorithm 1.

6 Discussion and future work

For a Gabidulin code we propose a transform–domain algorithm correcting both errors and erasures. We also give a generalization of the algorithm for interleaved Gabidulin codes.

Straightforward implementation of these algorithms have time complexity \(O(n^2)\) operations over \(\mathbb{F}_{q^m}\) and \(O(Ln^2)\) respectively. So, for fixed \(L\) both algorithms have quadratic in the code length complexity \(O(n^2)\). The order of complexity is the same as the one in [6], and is less than the one in [18], [19], which has order \(O(n^3)\).

The proposed algorithm have a potential to be accelerated and we are working in this direction. Fast methods for multiplication of linearized polynomials were found in [11] and [10]. Fast solution of the key equations were suggested in [15]. We need to find a fast method for finding a minimal power linearized polynomial, given a basis of its roots, and a fast method for left and right series power expansion or for left and right division of skew polynomials. After this we will have a fast decoding algorithm having complexity sub-quadratic in the code length.

Acknowledgements The authors would like to thank Antonia Wachter-Zeh for helpful discussions.

References

On transform–domain decoding of Gabidulin codes


18. P. Loidreau, R. Overbeck, “Decoding rank errors beyond the error-correction capability,” in Proc. 10th international workshop on algebraic and combinatorial coding theory, ACCT-10, Zvenigorod, Russia, Sept. 2006

Interpolation-Based Decoding of Interleaved Gabidulin Codes

Antonia Wachter-Zeh · Alexander Zeh

February 12, 2013

Abstract (Interleaved) Gabidulin codes are a special class of codes in rank metric and can be applied in random linear network coding and space-time coding. In this contribution, a new interpolation-based approach for decoding interleaved Gabidulin codes of interleaving order $s$ and length $n$ is considered. The decoding algorithm relies on constructing a multi-variate linearized polynomial, which interpolates the points $\{(\alpha_i, r_1^{(1)}(i), \ldots, r_s^{(s)}(i))\}_{i=0}^{n-1}$, where $r_1^{(1)}(i), \ldots, r_s^{(s)}(i)$ are the coefficients of the received words. We prove that the evaluation polynomials (of degree less than $k$) of any interleaved Gabidulin codeword in rank distance less than $s/(s+1)(n-k+1)$ are roots of the multi-variate polynomial. Moreover, we show how determining these roots can be done by simply solving a linear system of equations. Our decoder can be interpreted as either a list decoding algorithm or a unique decoding algorithm with a certain failure probability. To our knowledge, it is the first list decoding algorithm for interleaved Gabidulin codes. For the unique decoder, we show a connection to known unique decoding approaches, which provides an upper bound on the failure probability.

Keywords (Interleaved) Gabidulin codes · Rank metric codes · Interpolation

1 Definitions and Background

Let $q$ be a power of a prime, let $\mathbb{F}_q$ denote the finite field of order $q$ and let $\mathbb{F}_{q^m}$ be the extension field of degree $m$ over $\mathbb{F}_q$. Moreover, let $\mathbb{F}_{q^m}^{s \times n}$ denote the set of all $s \times n$ matrices over $\mathbb{F}_{q^m}$ and let $\mathbb{F}_{q^m}^{n} \overset{\text{def}}{=} \mathbb{F}_{q^m}^{1 \times n}$. Denote $[i] \overset{\text{def}}{=} q^i$ for any integer $i$.
For a vector $a = (a_0 \ a_1 \ldots \ a_{n-1})$ over $\mathbb{F}_q^m$, we denote the $s \times n$ Moore matrix by:

$$
\mathcal{M}_s(a) = \mathcal{M}_s(a_0 \ a_1 \ldots \ a_{n-1}) \equiv \begin{pmatrix}
a_0 & a_1 & \ldots & a_{n-1} \\
[\alpha] & [\alpha] & \ldots & [\alpha] \\
\vdots & \vdots & \ddots & \vdots \\
a_0^{[s-1]} & a_1^{[s-1]} & \ldots & a_{n-1}^{[s-1]}
\end{pmatrix}.
$$

If $a_0, \ldots, a_{n-1} \in \mathbb{F}_q^m$ are linearly independent over $\mathbb{F}_q$, $\text{rk}(\mathcal{M}_s(a)) = \min(s, n)$.

Given a basis of $\mathbb{F}_q^m$ over $\mathbb{F}_q$, there exists a one-to-one mapping for each vector $x \in \mathbb{F}_q^m$ to a matrix $X \in \mathbb{F}_q^{n \times m}$. Denote by $\text{rk}(x)$ the (usual) rank of $X$ over $\mathbb{F}_q$ and let $R_q(x) = R_q(X)$ denote the row space of $X$ over $\mathbb{F}_q$. When it is not clear whether we consider the rank over $\mathbb{F}_q$ or $\mathbb{F}_q^m$, the rank over $\mathbb{F}_q$ is denoted by $\text{rk}(\cdot)$.

We use the notation as vector or matrix equivalently, whatever is more convenient.

Linearized polynomials were introduced by Ore [1]. Over $\mathbb{F}_q^m$ they have the form $f(x) = \sum_{i=0}^{d_f} f_i x^i$, with $f_i \in \mathbb{F}_q^m$. If $d_f \neq 0$, we call $d_f = \deg_q(f(x))$ the $q$-degree of $f(x)$. For all $\alpha, \beta \in \mathbb{F}_q$ and all $a, b \in \mathbb{F}_q^m$, the following holds: $f(\alpha a + \beta b) = \alpha f(a) + \beta f(b)$. The (usual) addition and the composition $f(g(x))$ convert the set of linearized polynomials into a non-commutative ring with identity element $x^{[0]} = x$.

In the following, all polynomials are linearized polynomials.

The minimum rank distance of a linear block code $\mathcal{C}$ over $\mathbb{F}_q^m$ is defined by

$$
d = \min \{	ext{rk}(c) \mid c \in \mathcal{C}, c \neq 0\}.
$$

A (vertically) interleaved Gabidulin code is defined as follows.

**Definition 1 (Interleaved Gabidulin Code)** A linear interleaved Gabidulin code $\mathcal{I}G(n, k_1, \ldots, k_s; s)$ over $\mathbb{F}_q^m$ of length $n \leq m$, elementary dimensions $k_1, \ldots, k_s$ and interleaving order $s$ is defined by:

$$
\mathcal{I}G(n, k_1, \ldots, k_s; s) \equiv \begin{pmatrix}
e^{(1)} \\
e^{(2)} \\
\vdots \\
e^{(s)}
\end{pmatrix} = \begin{pmatrix}
\begin{pmatrix} f^{(1)}(a_0) f^{(1)}(a_1) \ldots f^{(1)}(a_{n-1}) \\
f^{(2)}(a_0) f^{(2)}(a_1) \ldots f^{(2)}(a_{n-1}) \\
\vdots \\
f^{(s)}(a_0) f^{(s)}(a_1) \ldots f^{(s)}(a_{n-1})
\end{pmatrix}
\end{pmatrix},
$$

where the linearized evaluation polynomials are $f^{(1)}(x) \in \mathbb{F}_q^m[x], \deg_q f^{(1)}(x) < k_i$, $k_i < n$ for $i = 1, \ldots, s$ and $a_0, \ldots, a_{n-1} \in \mathbb{F}_q^m$ are linearly independent over $\mathbb{F}_q$.

For $s = 1$, this defines a usual Gabidulin code $\mathcal{G}(n, k) = \mathcal{I}G(n, k; 1)$, see also [2].

Let $r^{(i)} = (r_0^{(i)} \ldots r_{n-1}^{(i)}) = e^{(i)} + e^{(i)}$ denote the $i$-th elementary received word for $i = 1, \ldots, s$. Moreover, let $C_q, E_q$ and $R_q$ denote the representations of $(c^{(1)}T \ldots c^{(s)}T)^T$, $(e^{(1)}T \ldots e^{(s)}T)^T$ and $(r^{(1)}T \ldots r^{(s)}T)^T$ as $(sm \times n)$ matrices over $\mathbb{F}_q$. We assume that every error matrix $E_q$ of rank $t$ is equi-probable.

In [3], Loidreau and Overbeck showed unique decoding of interleaved Gabidulin codes (with $k_i = k$ for $i = 1, \ldots, s$) up to $\tau = s/(s+1)(n-k)$ errors with high probability. They solve a linear system of equations $R_{LO} \cdot \lambda = 0$, where the $(n - \tau - 1 + s(n - \tau - k)) \times n$ matrix $R_{LO}$ depends on $a_0, \ldots, a_{n-1}$ and the received words, and upper bound the probability that $\text{rk}(R_{LO}) < n - 1$ by $\frac{1}{q^m}$ when $\tau \geq s$. 

Let us denote the coefficients of the polynomials by \( \alpha \) and \( q \). Problem 1 with \( \alpha \) and \( q \) is a non-zero interpolation polynomial of the form \( Q(x) = Q_0(x) + Q_1(y_1) + \cdots + Q_s(y_s) \), which fulfills:

- \( Q(\alpha_i, r_i^{(1)}, \ldots, r_i^{(s)}) = 0 \), for \( i = 0, \ldots, n - 1 \),
- \( \deg_q Q_0(x) < n - \tau \), \( \deg_q Q_i(y_i) < n - \tau - (k_i - 1) \), for \( i = 1, \ldots, s \),

where \( \alpha_0, \ldots, \alpha_{n-1} \) are the evaluation points of the code.

Let us denote the coefficients of the polynomials by \( Q_0(x) = \sum_{j=0}^{n-\tau-1} q_{0,j} x^j \) and \( Q_0(x) = \sum_{j=0}^{n-1} q_{1,j} x^j \) for \( i = 1, \ldots, s \). The system of equations for solving Problem 1 with \( \alpha = (\alpha_0 \ldots \alpha_{n-1}) \), will be denoted by \( I \cdot \mathbf{q} = \mathbf{0} \), where

\[
I = \left[ M_{n-\tau} \mathbf{r}_1^T \ M_{n-\tau-k_1+1} \mathbf{r}_1^{(1)}^T \ \cdots \ M_{n-\tau-k_s+1} \mathbf{r}_s^{(s)}^T \right],
\]

and \( \mathbf{q}^T = (q_{0,0} \cdots q_{0,n-\tau-1} \ | \ q_{1,0} \cdots q_{1,n-\tau-k_1} \ | \ \cdots \ | q_{s,0} \cdots q_{s,n-\tau-k_s}) \).

**Lemma 1** There exists a non-zero interpolation polynomial \( Q(x, y_1, \ldots, y_s) \), fulfilling the conditions of Problem 1 if

\[
\tau < \frac{sn + s - \sum_{i=1}^{s} k_i}{s + 1}.
\]

**Proof** The number of linearly independent equations is at most \( n \) and has to be less than the number of unknowns, hence:

\[
n < n - \tau + \sum_{i=1}^{s} (n - \tau - k_i + 1) \iff \tau(s + 1) < sn + s - \sum_{i=1}^{s} k_i.
\]

For the special case \( k_i = k \) for \( i = 1, \ldots, s \), this gives \( \tau < s/(s+1)(n - k + 1) \).

The following theorem shows that the evaluation words of the interleaved Gabidulin code are a root of the interpolation polynomial of Problem 1.

2 Interpolation-Based Decoding of Interleaved Gabidulin Codes

As [6, 7], our decoding principle consists of an interpolation and a root-finding step. First, we give interpolation constraints for a multi-variate linearized interpolation polynomial \( Q(x, y_1, \ldots, y_s) \) and prove that the evaluation polynomials \( f^{(1)}(x), \ldots, f^{(s)}(x) \) are roots of \( Q(x, y_1, \ldots, y_s) \) up to a certain \( \tau \). Second, we show how the root finding step can be accomplished by solving a linear system of equations. An analysis what happens if the list size is greater than one and/or the system of equations for the root-finding step does not have full rank follows later.

2.1 Interpolation Step

**Problem 1 (Interpolation Step)** Find a non-zero \((s + 1)\)-variate linearized polynomial of the form \( Q(x, y_1, \ldots, y_s) = Q_0(x) + Q_1(y_1) + \cdots + Q_s(y_s) \), which fulfills:

- \( Q(\alpha_i, r_i^{(1)}, \ldots, r_i^{(s)}) = 0 \), for \( i = 0, \ldots, n - 1 \),
- \( \deg_q Q_0(x) < n - \tau \), \( \deg_q Q_i(y_i) < n - \tau - (k_i - 1) \), for \( i = 1, \ldots, s \),

where \( \alpha_0, \ldots, \alpha_{n-1} \) are the evaluation points of the code.

The following theorem shows that the evaluation words of the interleaved Gabidulin code are a root of the interpolation polynomial of Problem 1.
Theorem 1 Let \( c^{(i)} = (f^{(i)}(\alpha_0) \ldots f^{(i)}(\alpha_{n-1})) \) for \( i = 1, \ldots, s \), where \( \deg_q f^{(i)}(x) < k_i \) and let \( r^{(i)} = c^{(i)} + e^{(i)} \). Assume, the rank of \( E_q \in \mathbb{F}_q^{sn \times n} \) is at most \( \tau \), where \( \tau \) is restricted as in (2). Let a non-zero multi-variate linearized polynomial \( Q(x_1, \ldots, y_s) \) be given, fulfilling the interpolation constraints from Problem 1. Then, 
\[
P(x) \overset{\text{def}}{=} Q \left( x, f^{(1)}(x), \ldots, f^{(s)}(x) \right) = 0.
\]

Proof Define \( R^{(i)}(x) \) and \( E^{(i)}(x) \) such that \( r^{(i)}_j = R^{(i)}(\alpha_j) \) and \( e^{(i)}_j = r^{(i)}_j - c^{(i)}_j = E^{(i)}(\alpha_j) \) for \( j = 0, \ldots, n - 1 \) and \( i = 1, \ldots, s \). Further, denote \( T(x) \overset{\text{def}}{=} Q(x, R^{(1)}(x), \ldots, R^{(s)}(x)) \). Since all polynomials are linearized, \( T(x) - P(x) = Q \left( 0, E^{(1)}(x), \ldots, E^{(s)}(x) \right) = Q_1 \left( E^{(1)}(x) \right) + Q_2 \left( E^{(2)}(x) \right) + \cdots + Q_s \left( E^{(s)}(x) \right) \). Then, 
\[
\left( T(\alpha_0) T(\alpha_1) \ldots T(\alpha_{n-1}) \right) - \left( P(\alpha_0) P(\alpha_1) \ldots P(\alpha_{n-1}) \right) = 
\left( \sum_{i=1}^{s} Q_1(e^{(i)}_1) \right) \left( \sum_{i=1}^{s} Q_1(e^{(i)}_2) \right) \left( \sum_{i=1}^{s} Q_1(e^{(i)}_{n-1}) \right).
\]

which we denote as \( m \times n \) matrices over \( \mathbb{F}_q \) by \( T_q - P_q = Q_q \). Due to Lemma 9 in the appendix, for the row spaces we have \( R(Q_q) \subseteq R(E_q) \). Because of the interpolation constraints, \( T_q = 0 \) and hence \( \text{rk}(P_q) = \text{rk}(Q_q) \leq \text{rk}(E_q) \leq \tau \). If \( \text{rk}(P_q) \leq \tau \), then the dimension of the root space of \( P(x) \) (see (3)) over \( \mathbb{F}_q^m \) is at least \( n - \tau \), which is only possible if its \( q \)-degree is at least \( n - \tau \). However, \( \deg_q P(x) \leq n - \tau - 1 \) due to the degree constraints and therefore \( P(x) = 0 \). \( \square \)

Instead of Gaussian elimination for solving the linear system of equations, for this step, the efficient interpolation from [8] can be used and the complexity of the interpolation step is then \( O(s^2 n(n - \tau)) \) operations over \( \mathbb{F}_q^m \).

2.2 Root-Finding Step

Similar to [9] for folded/derivative Reed–Solomon codes and to [10] for folded Gabidulin codes, the root-finding step of our approach results in a linear system of equations in the coefficients of \( f^{(1)}(x), \ldots, f^{(s)}(x) \). Let the list size \( \ell \) denote the number of codewords of \( \mathcal{G}(n, k_1, \ldots, k_s; s) \) in rank distance at most \( \tau \) from \( R_q \).

Let \( Q(x, y_1, \ldots, y_s) \) be given. Then, the task of the root-finding step is to find all tuples of polynomials \( f^{(1)}(x), \ldots, f^{(s)}(x) \) such that \( P(x) = 0 \) (see (3)). The important observation is that this is a linear system of equations over \( \mathbb{F}_q^m \) in the coefficients of \( f^{(1)}(x), \ldots, f^{(s)}(x) \). Note for this purpose that \( (a + b)^{[i]} = a^{[i]} + b^{[i]} \) for any \( a, b \in \mathbb{F}_q^m \) and any integer \( i \).

Example 1 (Root-Finding) Let \( s = 2, n = m = 7, k_1 = k_2 = k = 2 \) and \( \tau = 3 \). We want to find all pairs \( f^{(1)}(x), f^{(2)}(x) \) such that \( P(x) = P_0 x^{[0]} + P_1 x^{[1]} + \cdots + P_{n-\tau-1} x^{[n-\tau-1]} = 0 \). Due to the constraints of Problem 1, \( \deg_q P(x) \leq n - \tau - 1 \) and:

\[
P_0 = 0 = q_0.0 + q_1.0 f_0^{(1)} + q_2.0 f_0^{(2)}
\]

\[
P_1 = 0 = q_0.1 + q_1.1 f_0^{(1)} + q_1.0 f_1^{(1)} + q_2.1 f_0^{(2)}[1] + q_2.0 f_1^{(2)}
\]

\[
\vdots
\]

\[
P_{n-\tau-1} = 0 = q_0.n-\tau-1 + q_1.n-\tau-1 f_0^{(1)}[n-\tau-k] + q_2.n-\tau-1 f_0^{(2)}[n-\tau-k].
\]
Therefore, given $Q(x, y_1, y_2)$, we can calculate the coefficients of all possible pairs $f^{(1)}(x)$, $f^{(2)}(x)$ by the following linear system of equations:

$$
\begin{pmatrix}
q_1 \cdot 0 & q_2 \cdot 0 \\
q_1 \cdot [-1] & q_2 \cdot [-1] \\
q_1 \cdot 2 & q_2 \cdot 2 \\
q_1 \cdot 1 & q_2 \cdot 1 \\
q_1 \cdot [-2] & q_2 \cdot [-2]
\end{pmatrix} \cdot \begin{pmatrix}
f^{(1)}_0 \\
f^{(1)}_1 \\
q^{(1)}_1[-1] \\
f^{(2)}_0 \\
q^{(2)}_1[-1]
\end{pmatrix} = \begin{pmatrix}
-\tau_0 \cdot 0 \\
-\tau_0 \cdot [-1] \\
-\tau_0 \cdot 1 \\
-\tau_0 \cdot 2 \\
-\tau_0 \cdot [-3] \\
-\tau_0 \cdot 3
\end{pmatrix}.
$$

(4)

In order to set up (4) in general, we can use more than one $Q(x, y_1, \ldots, y_s)$, namely we can take all polynomials corresponding to a basis vector of the solution space of the interpolation step. This also decreases the probability that the system of equations for the root-finding step does not have full rank (see also Section 4). To calculate the dimension of the solution space of the interpolation step, denoted by $d_I$, we need the rank of the interpolation matrix.

**Lemma 2** Let $\tau$ be as in (2) and let the rank of the error matrix $E_q$ over $\mathbb{F}_q$ be $t \leq \tau$. Then, for the interpolation matrix from (1), $\text{rk}(I) = n - \tau + t$.

**Proof** The first $k_i$ columns of $I$ contain the (transposed) generator matrices of the Gabidulin codes $G(n, k_i)$. Hence, for calculating the rank of $I$, we can subtract the codewords and their Frobenius powers from the $s$ right submatrices such that these submatrices only depend on the error. Hence, the rank of $I$ depends on $\text{rk}(M_{n-\tau}(\alpha))$, which is $n - \tau$ and on the rank of the error matrix, which is $t$. Since the errors are linearly independent of $(M_{n-\tau}(\alpha)^T$, $\text{rk}(I) = n - \tau + t$.

The dimension of the solution space of the interpolation step is therefore:

$$
\text{dim}(I) = (s + 1)(n - \tau) - \sum_{i=0}^{s}(k_i - 1) - (n - \tau + t).
$$

and for $k_i = k$ for $i = 1, \ldots, s$, we obtain $d_I = s(n - \tau - k + 1) - t$.

In the following, let $Q^{(h)}(x, y_1, \ldots, y_s)$ for $h = 1, \ldots, d_I$ denote the interpolation polynomials corresponding to the different basis vectors of the solution space of the interpolation step. Let us denote the following matrices:

$$
Q^{(h)}_j \overset{\text{def}}{=} \begin{pmatrix}
q^{(1)(j)} \cdot [1] & q^{(2)(j)} \cdot [1] & \cdots & q^{(s)(j)} \cdot [1] \\
q^{(1)(j)} \cdot 1 & q^{(2)(j)} \cdot 1 & \cdots & q^{(s)(j)} \cdot 1 \\
q^{(1)(j)} \cdot [-2] & q^{(2)(j)} \cdot [-2] & \cdots & q^{(s)(j)} \cdot [-2] \\
q^{(1)(j)} \cdot 2 & q^{(2)(j)} \cdot 2 & \cdots & q^{(s)(j)} \cdot 2 \\
q^{(1)(j)} \cdot 1 & q^{(2)(j)} \cdot 1 & \cdots & q^{(s)(j)} \cdot 1 \\
& \vdots & & \vdots \\
& q^{(1)(j)} \cdot (d_I) & q^{(2)(j)} \cdot (d_I) & \cdots & q^{(s)(j)} \cdot (d_I)
\end{pmatrix},
$$

$$
J^{(h)}_j \overset{\text{def}}{=} \begin{pmatrix}
f^{(1)(j)} \cdot [1] \\
f^{(2)(j)} \cdot [1] \\
\vdots \\
f^{(1)(j)} \cdot s \\
f^{(2)(j)} \cdot s \\
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{pmatrix},
$$

$$
Q^{(h)}_j \overset{\text{def}}{=} \begin{pmatrix}
q^{(1)(j)} \cdot [1] \\
q^{(2)(j)} \cdot [1] \\
\vdots \\
q^{(1)(j)} \cdot (d_I) \\
q^{(2)(j)} \cdot (d_I) \\
\vdots \\
\vdots \\
\vdots \\
\vdots
\end{pmatrix}.
$$

(5)
For $k = \max_i \{k_i\}$, the linear system of equations for the root-finding step is:
\[
Q^{(h)}(x, f^{(1)}(x), \ldots, f^{(s)}(x)) = Q_0^{(h)}(x) + Q_1^{(h)}(f^{(1)}(x)) + \cdots + Q_s^{(h)}(f^{(s)}(x)) = 0,
\]
for all $h = 1, \ldots, d_I$,
\[
\begin{pmatrix}
Q_0^{[-1]} & Q_1^{[-1]} & Q_2^{[-2]} & Q_0^{[-2]} \\
\cdots & \cdots & \cdots & \cdots \\
Q_{n-\tau-k}^{[-(n-\tau-3)]} & Q_{n-\tau-k}^{[-(n-\tau-3)]} & Q_{n-\tau-k}^{[-(n-\tau-3)]} & \cdots \\
Q_{n-\tau-k}^{[-(n-\tau-3)]} & Q_{n-\tau-k}^{[-(n-\tau-2)]} & Q_{n-\tau-k}^{[-(n-\tau-2)]} & \cdots \\
Q_{n-\tau-k}^{[-(n-\tau-2)]} & Q_{n-\tau-k}^{[-(n-\tau-1)]} & Q_{n-\tau-k}^{[-(n-\tau-1)]} & \cdots \\
\end{pmatrix}
\begin{pmatrix}
f_0 \\
f_1 \\
\vdots \\
f_{k-1}
\end{pmatrix}
= \begin{pmatrix}
-q_0^{[-1]} \\
-q_1^{[-1]} \\
\vdots \\
-q_{0,n-\tau-1}^{[-(n-\tau-1)]}
\end{pmatrix},
\]
which we denote by $Q \cdot f = q_0$, where $Q$ is a $((n - \tau)d_I) \times sk$ matrix. For arbitrary elementary dimension $k_1, \ldots, k_s$, the forming of the system of equations for the root-finding step is straight-forward.

**Lemma 3 (Complexity of the Root-Finding Step)** Let $Q^{(h)}(x, y_1, \ldots, y_s)$ for $h = 1, \ldots, d_I$ be given. Then, the basis of the affine subspace containing the coefficients of all tuples $f^{(1)}(x), \ldots, f^{(s)}(x)$ such that $P(x) = 0$ can be found recursively with complexity at most $O(s^3k^2)$ operations over $\mathbb{F}_{q^n}$.

The proof is omitted due to space restrictions.

### 3 Interpretation as List Decoding Algorithm

Our decoding approach for interleaved Gabidulin codes can be seen as a list decoding algorithm, consisting of solving two linear systems of equations. Except for pruning the solution space of the root-finding step, we find the solution(s) with complexity at most $O(s^3n^2)$. For the interpretation as a list decoding algorithm, we analyze the maximum and average list size in the following.

**Lemma 4 (Maximum List Size)** Let $c^{(i)} = (f^{(i)}(a_0), \ldots, f^{(i)}(a_{n-1}))$ for $i = 1, \ldots, s$, where $\deg a f^{(i)}(x) < k_i$ and let $r^{(i)} = c^{(i)} + e^{(i)}$. Let $\text{rk}(E_q) = t \leq \tau$ and let $\tau$ be as in (2). Then, the list size $\ell$, i.e., the number of codewords from $I\hat{G}(n, k_1, \ldots, k_s; s)$ in rank distance less or equal $\tau$ to the received word, is upper bounded by
\[
\ell \leq q^{m(\sum_{i=1}^s k_i - \min_i(k_i))}.
\]

The proof is omitted due to space restrictions.

However, it is not clear whether the list size $\ell$ can really be that great. Moreover, finding the actual list out of the solution space of (6) reduces the list size.

When $\ell > 1$, the system of equations for the root-finding step (6) cannot have full rank. The following lemma estimates the average list size. For most parameters, this value is almost one (see Example 2). This proof proceeds similar to [11].
Lemma 5 (Average List Size) Let \( c^{(i)} = (f^{(i)}(\alpha_0) \ldots f^{(i)}(\alpha_{n-1})) \) for \( i = 1, \ldots, s \), where \( \deg_q f^{(i)}(x) < k_i \) and let \( r^{(i)} = c^{(i)} + e^{(i)} \). Let \( \text{rk}(E_q) \leq \tau \) and let \( \tau \) be as in (2). Then, the average list size, i.e., the number of codewords \( C_q \) from \( IG(n, k_1, \ldots, k_s; s) \) such that \( \text{rk}(R_q - C_q) \leq \tau \) is upper bounded by

\[
\bar{\ell} < 1 + 4 \left( q^{m} \sum_{i=1}^{s} k_i - 1 \right) q^{(sn+n)\tau - \tau^2 - smn},
\]

and for \( k_i = k \) for \( i = 1, \ldots, s \), the bound is

\[
\bar{\ell} < 1 + 4(q^{smk} - 1)q^{(sn+n)\tau - \tau^2 - smn}.
\]

Proof Let \( R \) be a random variable, uniformly distributed over all matrices in \( F_q^{n \times \tau} \) and let \( r \) be a realization of \( R \), i.e., the \( s \) elementary received words written as rows of a matrix. Let \( c \in IG(n, k_1, \ldots, k_s; s) \) be the fixed transmitted codeword.

Then, \( P(\text{rk}(r - c) \leq \tau) = P(\text{rk}(r) \leq \tau) \), which is the probability that a random \( sm \times n \) matrix over \( F_q \) has rank at most \( \tau \). Let \( IG^*(n, k_1, \ldots, k_s; s) \) be the code \( IG(n, k_1, \ldots, k_s; s) \) without the transmitted codeword.

Let us further consider another random variable, which depends on \( R \):

\[
X(R) = |\{c \in IG^* : \text{rk}(r - c) \leq \tau\}|.
\]

Let \( 1(\cdot) \) denote the indicator function, then the expectancy can be calculated by:

\[
E[X] = \sum_{r \in F_q^{n \times \tau}} \sum_{c \in IG^*} 1(\text{rk}(r - c) \leq \tau) P(R = r)
\]

\[
= \sum_{c \in IG^*} E[1(\text{rk}(r - c) \leq \tau)] = \sum_{c \in IG^*} P(\text{rk}(r) \leq \tau) = \sum_{c \in IG^*} P(\text{rk}(r) \leq \tau)
\]

\[
= |IG^*| \sum_{R_q \in F_q^{m \times n}} q^{m} \sum_{k_i = 1}^{\tau} \left( q^{m} \sum_{i=1}^{s} k_i - 1 \right) q^{(sm+n)\tau - \tau^2 - smn},
\]

and hence for \( k_i = k \) for \( i = 1, \ldots, s \), we obtain \( E[X] < 4(q^{smk} - 1)q^{(sn+n)\tau - \tau^2 - smn} \).

The average list size is \( \bar{\ell} = E[X] + 1 \), since we add the transmitted codeword. \( \square \)

Unfortunately, it is not clear if it is possible that \( \ell = 1 \), but the system of equations for the root-finding step (6) does not have full rank. Thus, Lemma 5 does not bound the probability that the rank of \( Q \) is not full; this will be done in Section 4.

Theorem 2 summarizes the properties of our list decoding algorithm.

Theorem 2 (List Decoding of Interleaved Gabidulin Codes) Let the interleaved Gabidulin code \( IG(n, k_1, \ldots, k_s; s) \) consist of the elementary codewords \( c^{(i)} = (f^{(i)}(\alpha_0) \ldots f^{(i)}(\alpha_{n-1})) \) for \( i = 1, \ldots, s \), where \( \deg_q f^{(i)}(x) < k_i \) and let the elementary received words \( r^{(i)} = c^{(i)} + e^{(i)} \) for \( i = 1, \ldots, s \) be given.

Then, we can find a basis of the subspace which contains all tuples of polynomials \( f^{(1)}(x), \ldots, f^{(s)}(x) \) such that their evaluation is in rank distance

\[
\tau < \frac{sn + s - \sum_{i=1}^{s} k_i}{s + 1}
\]

to the received word with overall complexity at most \( O(s^3 n^2) \).
4 Interpretation as Probabilistic Unique Decoding Algorithm

In this section, we consider our decoding approach as a probabilistic unique decoding algorithm. Since the list size might be greater than one, there is not always a unique solution. We declare a decoding failure as soon as the rank of $Q$ (see (6)) is not full. We upper bound this probability and call it failure probability. Moreover, we show a relation to the unique decoding approach from [3]. The upper bound as well as simulation results show that the failure probability is quite small.

For simplicity, we consider only $k_i = k$ for all $i = 1, \ldots, s$, but the extension to arbitrary $k_i$ is straight-forward. Recall the notations from (5) and denote additionally the $d_I \times (s + 1)$ matrix

$$Q_0 \overset{\text{def}}{=} \begin{pmatrix} q_{0,0}^{(1)} & q_{1,0}^{(1)} & \cdots & q_{s,0}^{(1)} \\ \vdots & \vdots & & \vdots \\ q_{0,0}^{(d_I)} & q_{1,0}^{(d_I)} & \cdots & q_{s,0}^{(d_I)} \end{pmatrix}.$$  

Note that $\text{rk}(Q_0^{[0]}) = \text{rk}(Q_0^{[i]})$ holds for any integer $i$ since the Frobenius power on the whole matrix is a linear operation. The matrix $Q$ of the root finding step (6) is a lower block triangular matrix and has $Q_0^{[0]}, Q_0^{[1]}, \ldots, Q_0^{[k-1]}$ on the diagonal of the first $k$ blocks (i.e., first $sk$ lines). This gives the following lemma.

**Lemma 6** If $\text{rk}(Q_0) = s$, then $\text{rk}(Q) = sk$.

**Proof** This holds since $Q$ is a lower block triangular matrix with $Q_0^{[0]}, \ldots, Q_0^{[k-1]}$ on the diagonal of the first $k$ blocks and since $\text{rk}(Q_0^{[0]}) = \text{rk}(Q_0^{[i]})$. $\square$

Clearly, the matrix $Q_0$ can have rank $s$ if and only if $d_I \geq s$, which is guaranteed for $t = \tau$ if:

$$s(n - k + 1) - (s + 1)\tau \geq s \iff \tau \leq s/(s+1)(n - k). \quad (7)$$

This is equivalent to the decoding radius of joint interleaved decoding and slightly different to (2), which is the maximum decoding radius when we consider our algorithm as a list decoder in Section 3.

In the following, we will show a connection between the probability that $Q$ does not have full rank and that the matrix $R_{LO}$ from [3] does not have full rank.

**Lemma 7** Let $\tau = \lceil s/(s+1)(n - k) \rceil$. If $\text{rk}(Q_0) < s$, then $\text{rk}(R_{LO}) < n - 1$, where $R_{LO}$ denotes the decoding matrix of the Loidreau–Overbeck approach [3].

**Proof** If $\text{rk}(Q_0) < s$, then by linearly combining the $d_I \geq s$ dimensional basis of the solution space of the interpolation step, there exists a non-zero interpolation polynomial $Q(x, y_1, \ldots, y_s)$, which fulfills Problem 1 and has the coefficients $q_{0,0} = q_{1,0} = \cdots = q_{s,0} = 0$. Since $Q(x, y_1, \ldots, y_s) \neq 0$ (Lemma 1), the interpolation matrix without the first column of each submatrix (i.e., the columns corresponding to $q_{0,0}, q_{1,0}, \ldots, q_{s,0}$), denoted by $I$, does not have full rank. Moreover $R_{LO}^{[i]} = I^T$ and hence, 

$$\text{rk}(R_{LO}) = \text{rk}(I) < \sum_{i=0}^{s} \deg_y Q_i(x) = (s + 1)(n - \tau) - sk - 1.$$ 

For $\tau = \lceil s/(s+1)(n - k) \rceil$, this gives $\text{rk}(R_{LO}) < n - 1$. $\square$
Combining these two lemmas, we obtain the following theorem.

**Theorem 3** Let \( \tau = \lfloor s/(s+1)(n-k) \rfloor \) and assume that the received words \( r^{(1)}, \ldots, r^{(s)} \) consist of random elements uniformly distributed over \( \mathbb{F}_q^n \). Then,

\[
P(\text{rk}(Q) < sk) \leq P(\text{rk}(\mathbf{Q}_0) < s) \leq P(\text{rk}(\mathbf{R}_{LO}) < n-1). \tag{8}
\]

Therefore, for \( \tau \geq s \):

\[
P(\text{rk}(Q) < sk) \leq P(\text{rk}(\mathbf{R}_{LO}) < n-1) \leq 4q^{-m}.
\]

**Proof** Since \( \tau = \lfloor s/(s+1)(n-k) \rfloor \), we have \( d_j = s \) and hence, \( \text{rk}(\mathbf{Q}_0) = \text{rk}(Q_0) \). The first inequality of (8) follows from Lemma 6 and the second from Lemma 7. Hence, we can bound \( P(t_R > 1) \) by the failure probability from [3]. \( \square \)

Note that the assumption of random received vectors and the restriction \( \tau \geq s \) follow from [12, Theorem 3.1]. We conjecture that \( \tau \geq s \) is only a technical restriction and that the results hold equivalently for \( \tau < s \).

Alternatively, we can bound the failure probability as follows. Assume, the matrix \( Q_0 \) consists of random values over \( \mathbb{F}_q^n \). This assumption seems to be reasonable, since in [3] and [4] it is assumed that \( r^{(1)}, \ldots, r^{(s)} \) are random vectors over \( \mathbb{F}_q^n \). In our approach, the values of \( Q_0 \) are obtained from a linear system of equations, where each \( q_{ij} \) is multiplied with the coefficients of a different \( r^{(i)} \).

**Lemma 8** Let the number of errors \( t \leq \tau \), where \( \tau = \lfloor s/(s+1)(n-k) \rfloor \) and let \( q_{1,0}^{(j)}, q_{2,0}^{(j)}, \ldots, q_{s,0}^{(j)} \) for \( j = 1, \ldots, d_j \) be random elements uniformly distributed over \( \mathbb{F}_q^n \). Then,

\[
P(\text{rk}(Q) < sk) \leq \frac{4}{q^{m(d_{t+1}-s)}} = 4q^{-m(s(n-k-\tau)-t+1)}.
\]

**Proof** It holds that \( d_j \geq s \) and hence with Lemma 6, if \( \text{rk}(Q_0) = s \), then \( \text{rk}(Q) = sk \). Hence, \( P(\text{rk}(Q) < sk) \leq P(\text{rk}(Q_0) < s) \). When \( q_{1,0}^{(j)}, \ldots, q_{s,0}^{(j)} \) for \( j = 1, \ldots, d_j \) are random elements from \( \mathbb{F}_q^n \), we can bound \( P(\text{rk}(Q_0) < s) \) by the probability that a random \((d_j \times s)\)-matrix over \( \mathbb{F}_q^n \) has rank less than \( s \):

\[
P(\text{rk}(Q) < sk) \leq P(\text{rk}(Q_0) < s) \leq \frac{\sum_{j=0}^{s-1} \prod_{h=0}^{j-1} q^{d_j - q^h} \prod_{k=0}^{j-1} (q^h - q)}{q^{msd_{t+1}}} = 4q^{-m(s(n-k-\tau)-t+1)}. \tag{8}
\]

Note that Lemma 8 does not have the technical restriction \( \tau \geq s \) as Theorem 3 and the bounds from [3,4]. The following theorem summarizes these results.

**Theorem 4** (Unique Decoding of Interleaved Gabidulin Codes) Let the interleaved Gabidulin code \( \mathcal{I}\mathcal{G}(n, k_1, \ldots, k_s; s) \) consist of the elementary codewords \( c^{(i)} = (f^{(i)}(\alpha_1) \ldots f^{(i)}(\alpha_{n-1})) \) for \( i = 1, \ldots, s \), where \( \deg f^{(i)}(x) < k_i \) and let the elementary received words \( r^{(i)} = c^{(i)} + e^{(i)} \) for \( i = 1, \ldots, s \) be given. Then, with probability at least

\[
1 - 4q^{-m(s(n-k-\tau)-t+1)},
\]

we can find a unique solution \( f^{(1)}(x), \ldots, f^{(s)}(x) \) such that its evaluation is in rank distance

\[
t \leq \tau = \left\lfloor \frac{sn - \sum_{i=1}^{s} k_i}{s+1} \right\rfloor
\]

to the received word with overall complexity at most \( O(s^3n^2) \).
Example 2 (Failure Probabilities) Let us consider the $\mathcal{IG}(7, 2, 2; 2)$ code over $\mathbb{F}_2^7$. The maximum decoding radius (for our approach as well as for $[3, 4]$) according to (2) is $\tau = 3$ whereas a Bounded Minimum Distance decoder could correct only two errors. We simulated $10^7$ transmissions, where any error matrix $E_q \in \mathbb{F}_q^{m \times n}$ of rank $\tau = 3$ occurred with equal probability. The following simulated probabilities occurred:

$$P(\text{rk}(Q) < sk) = P(\text{rk}(S_{SB}) < \tau) = P(\text{rk}(R_{LO}) < n - 1) = 6.12 \cdot 10^{-5}.$$  

The average list size due to Lemma 5 is $\tilde{l} = 1 + 6.104 \cdot 10^{-5}$.

The upper bound from Theorem (3) (and therefore the upper bound from [3]) gives

$$P(\ell_R > 1) \leq P(\text{rk}(R_{LO}) < n - 1) \leq 4q^{-m} = 0.0312,$$

and the bound from Lemma 8 gives

$$P(\ell_R > 1) \leq 4q^{-m(s(n-k-\tau)-\tau+1)} = 2.44 \cdot 10^{-4}.$$  

This example leads to the following conjecture.

Conjecture 1 We conjecture that if and only if $\text{rk}(Q) < sk$, then $\text{rk}(S_{SB}) < \tau$ and $\text{rk}(R_{LO}) < n - 1$. Hence, we conjecture that Lemma 7 holds in both directions.

5 Conclusion

We presented an approach for decoding interleaved Gabidulin codes based on interpolating a multi-variate linearized polynomial. The procedure consists of two steps: an interpolation step and a root-finding step, where both can be accomplished by solving a system of equations. This new decoder can be used as a list decoder for interleaved Gabidulin codes or as a unique decoder with a certain failure probability. The complexity of both approaches is quadratic in the length of the code and the output is a unique decoding result with high probability. The connection to known approaches for decoding interleaved Gabidulin codes and an upper bound on the failure probability are derived.

Appendix

Lemma 9 Let $A(x)$ and $E(x)$ denote two linearized polynomials over $\mathbb{F}_q^m$ with $\text{deg}_q A(x) \leq n \leq m$ and $\text{deg}_q E(x) \leq n \leq m$. Let $e = (e_0 \ldots e_{n-1}) = (E(\alpha_0) \ldots E(\alpha_{n-1}))$ and let $\mathcal{R}(e)$ denote the row space of $e$ represented as $m \times n$ matrix over $\mathbb{F}_q$. Then,

$$\mathcal{R}(A(e_0), \ldots, A(e_{n-1})) \subseteq \mathcal{R}(e).$$

The proof is omitted due to space restrictions.

Acknowledgment

The authors thank Vladimir Sidorenko for the valuable discussions and the reviewers for their suggestions that helped to improve the presentation of the paper.
References

List Decoding of Lifted Gabidulin Codes via the Plücker Embedding

Anna-Lena Trautmann · Natalia Silberstein · Joachim Rosenthal

Abstract Codes in the Grassmannian have recently found an application in random network coding. All the codewords in such codes are subspaces of \( \mathbb{F}_q^n \) with a given dimension.

In this paper, we consider the problem of list decoding of a certain family of codes in the Grassmannian, called lifted Gabidulin codes. For this purpose we use the Plücker embedding of the Grassmannian. We describe a way of representing a subset of the Plücker coordinates of lifted Gabidulin codes as linear block codes. The union of the parity-check equations of these block codes and the equations which arise from the description of a ball around a subspace in the Plücker coordinates describe the list of codewords with distance less than a given parameter from the received word.

1 Introduction

Let \( \mathbb{F}_q \) be a finite field of size \( q \). The Grassmannian space (Grassmannian, in short), denoted by \( G_q(k, n) \), is the set of all \( k \)-dimensional subspaces of the vector space \( \mathbb{F}_q^n \), for any given two integers \( k \) and \( n \), \( 0 \leq k \leq n \). A subset \( C \) of the Grassmannian is called an \( (n, M, d_S, k)_q \) constant dimension code if it has size \( M \) and minimum subspace distance \( d_S \), where the distance function in \( G_q(k, n) \) is defined as follows:

\[
d_S(U, V) = 2k - 2 \dim(U \cap V),
\]

for any two subspaces \( U \) and \( V \) in \( G_q(k, n) \).

These codes gained a lot of interest due to the work by Kötter and Kschischang [13], where they showed the application of such codes for error-correction in random network coding. They proved that an \( (n, M, d_S, k)_q \) code can correct any \( t \) packet errors (which is equivalent to \( t \) packet insertions and \( t \) packet deletions) and any \( \tau \) packet erasures introduced anywhere in the network as long as \( 4t + 2\tau < d_S \). This application has motivated extensive work in the area [1,3,5,7–9,12, 14–16,18,21,24,25]. In [13] Kötter and Kschischang gave a Singleton like upper bound on the size of such codes and presented a Reed-Solomon like code which asymptotically attains this bound. Silva, Kötter, and Kschischang [20] showed how this construction can be described in terms of lifted Gabidulin codes [6]. The generalizations of this construction and the decoding algorithms...

First and third author were partially supported by Swiss National Science Foundation Grant no. 138080. The first author was partially supported by Forschungskredit of the University of Zurich, grant no. 57104103.

A.-L. Trautmann · J. Rosenthal
Institute of Mathematics, University of Zurich, Switzerland
E-mail: trautmann@math.uzh.ch, rosenthal@math.uzh.ch

N. Silberstein
Department of Electrical and Computer Engineering, University of Texas at Austin, USA
E-mail: natalys@austin.utexas.edu.
were presented in [1,3,14,18,21,25]. Another type of construction (orbit codes) can be found in [5, 12,24].

In this paper we focus on the list decoding of lifted Gabidulin codes. For the classical Gabidulin codes it was recently shown by Wachter-Zeh [26] that, if the radius of the ball around a received word is greater than the Johnson radius, no polynomial-time list decoding is possible, since the list size can be exponential. Algebraic list decoding algorithms for folded Gabidulin codes were discussed in [8,15]. The constructions of subcodes of lifted Gabidulin codes and their algebraic list decoding algorithms were presented in [9,16].

Our approach for list decoding codes in the Grassmannian is to apply the techniques of Schu- bert calculus over finite fields, i.e. we represent subspaces in the Grassmannian by their Plücker coordinates. It was proven in [18] that a ball of a given radius (with respect to the subspace distance) around a subspace can be described by explicit linear equations in the Plücker embedding. In this work we describe a way of representing a subset of the Plücker coordinates of lifted Gabidulin codes as linear block codes, which results in additional linear (parity-check) equations. The solutions of all these equations will constitute the resulting list of codewords.

The rest of this paper is organized as follows. In Section 2 we describe the construction of Gabidulin and lifted Gabidulin codes and discuss the Plücker embedding of subspaces in the Grassmannian. In Section 3 we describe a representation of a subset of the Plücker coordinates of a lifted Gabidulin code and present a list decoding algorithm. Conclusions and problems for future research are given in Section 4.

2 Preliminaries and Notations

We denote by $GL_n$ the general linear group over $\mathbb{F}_q$, by $S_n$ the symmetric group of degree $n$. With $\mathbb{P}^n$ we denote the projective space of order $n$ over $\mathbb{F}_q$.

Let $p(x) = \sum p_i x^i \in \mathbb{F}_q[x]$ be a monic and irreducible polynomial of degree $\ell$, and $\alpha$ be a root of $p(x)$. Then it holds that $\mathbb{F}_{q^\ell} \cong \mathbb{F}_q[\alpha]$. We denote the vector space isomorphism between the extension field $\mathbb{F}_{q^\ell}$ and the vector space $\mathbb{F}_q^\ell$ by

$$\phi: \mathbb{F}_{q^\ell} \rightarrow \mathbb{F}_q^\ell$$

$$\sum_{i=0}^{\ell-1} \lambda_i \alpha^i \mapsto (\lambda_0, \ldots, \lambda_{\ell-1}).$$

Moreover, we need the following notations: $rs(U)$ denotes the row space of a matrix $U$,

$$\binom{[n]}{k} := \{ (x_1, \ldots, x_k) \mid x_i \in \{1,2,\ldots,n\}, x_1 < \cdots < x_k \},$$

and for a matrix $A$ we denote its $i$-th row by $A[i]$, its $i$-th column by $A_i$, and the entry in the $i$-th row and the $j$-th column by $A_{i,j}$.

2.1 Lifted Gabidulin Codes

For two $k \times \ell$ matrices $A$ and $B$ over $\mathbb{F}_q$ the rank distance is defined by

$$d_R(A, B) \overset{\text{def}}{=} \operatorname{rank}(A - B).$$

A $[k \times \ell, \varrho, \delta]$ rank-metric code $C$ is a linear subspace with dimension $\varrho$ of $\mathbb{F}_q^{k \times \ell}$, in which each two distinct codewords $A$ and $B$ have distance $d_R(A, B) \geq \delta$. For a $[k \times \ell, \varrho, \delta]$ rank-metric code $C$ it was proven in [2,6,19] that

$$\varrho \leq \min\{ k(\ell - \delta + 1), \ell(k - \delta + 1) \}.$$
The codes which attain this bound are called \textit{maximum rank distance} codes (or MRD codes in short).

An important family of MRD linear codes was presented by Gabidulin [6]. These codes can be seen as the analogs of Reed-Solomon codes for the rank metric. From now on let \( k \leq \ell \). A codeword \( A \) in a \([k \times \ell, q, \delta]\) rank-metric code \( C \) can be represented by a vector \( c_A = (c_1, c_2, \ldots, c_k) \), where \( c_i = \phi_i(A[i]) \in F_q^\ell \). Let \( g_i \in F_q^\ell \), \( 1 \leq i \leq k \), be linearly independent over \( F_q \). Then the generator matrix \( G \) of a \([k \times \ell, q, \delta]\) Gabidulin MRD code is given by

\[
G = \begin{pmatrix}
g_1 & g_2 & \cdots & g_k \\
g_1^{[1]} & g_2^{[1]} & \cdots & g_k^{[1]} \\
g_1^{[2]} & g_2^{[2]} & \cdots & g_k^{[2]} \\
\vdots & \vdots & \ddots & \vdots \\
g_1^{[k-\delta]} & g_2^{[k-\delta]} & \cdots & g_k^{[k-\delta]}
\end{pmatrix},
\]

where \( \ell = \ell(k-\delta+1) \), and \([i] = q^i \). [6]

Let \( A \) be a \([k \times \ell]\) matrix over \( F_q \) and let \( I_k \) be the \([k \times k]\) identity matrix. The matrix \([I_k, A]\) can be viewed as a generator matrix of a \( k \)-dimensional subspace of \( F_q^{k+\ell} \). This subspace is called the \textit{lifting} of \( A \) [20].

When the codewords of a rank-metric code \( C \) are lifted to \( k \)-dimensional subspaces, the result is a constant dimension code \( C \). If \( C \) is a Gabidulin MRD code then \( C \) is called a \textit{lifted Gabidulin (LG)} code.

\textbf{Theorem 1} [20] Let \( k, n \) be positive integers such that \( k \leq n - k \). If \( C \) is a \([k \times (n-k), (n-k)(k-\delta+1), \delta]\) Gabidulin MRD code then \( C \) is an \((n, q^{(n-k)(k-\delta+1)}, 2\delta, k)_q\) constant dimension code.

\subsection{2.2 The Plücker Embedding}

The basic idea of using the Plücker embedding for list decoding of subspace codes was already stated in [18,23]. We will now recall the main definitions and theorems from those works. The proofs of the results can also be found in there. For more information or a more general formulation of the Plücker embedding and its applications the interested reader is referred to [10].

Let \( U \in \mathbb{F}_q^{mk \times n} \) such that its row space \( \text{rs}(U) \) describes the subspace \( U \in \mathcal{G}_q(k, n) \). \( M_{i_1, \ldots, i_k}(U) \) denotes the minor (i.e. the determinant of the submatrix) of \( U \) given by the columns \( i_1, \ldots, i_k \). The Grassmannian \( \mathcal{G}_q(k, n) \) can be embedded into projective space using the Plücker embedding:

\[
\varphi : \mathcal{G}_q(k, n) \longrightarrow \mathbb{P}^{\binom{n}{k}-1}
\]

\[
\text{rs}(U) \mapsto [M_{1, \ldots, k}(U) : M_{1, \ldots, k-1, k+1}(U) : \ldots : M_{n-\ell+1, \ldots, n}(U)].
\]

The \( k \times k \) minors \( M_{i_1, \ldots, i_k}(U) \) of the matrix \( U \) are called the \textit{Plücker coordinates} of the subspace \( U \). By convention, we order the minors lexicographically by the column indices. Recall, that the lexicographic order is defined as follows:

\[
(i_1, \ldots, i_k) < (j_1, \ldots, j_k) \iff \exists 0 \leq Nk : i_m = j_m \forall m < N \text{ and } i_{N+1} < j_{N+1}.
\]

The image of this embedding describes indeed a variety and the defining equations of the image are given by the so called \textit{shuffle relations} (see e.g. [11,17]), which are multilinear equations of monomial degree 2 in terms of the Plücker coordinates:

\textbf{Proposition 2} Consider \( x := [x_1, \ldots, k : \cdots : x_{n-k+1}, \ldots, n] \in \mathbb{P}^{\binom{n}{k}-1}. \) Then there exists a subspace \( U \in \mathcal{G}_q(k, n) \) such that \( \varphi(U) = x \) if and only if

\[
\sum_{j \in \{i_1, \ldots, i_{k+1}\}} \text{sgn}(\sigma_j)x_{i_1, \ldots, i_{k+1}\setminus j}x_{i_{k+2}, \ldots, i_{2k+1} \cup j} = 0 \quad \forall (i_1, \ldots, i_{2k}) \in \binom{[n]}{2k},
\]
where \( \sigma_j \) denotes the sign of the permutation such that

\[
\sigma_j(i_1, \ldots, i_{2k}) = (i_1, \ldots, i_{k+1} \setminus j) \upharpoonright [i_{k+2}, \ldots, i_{2k} \cup j).
\]

Then one can easily count the number of different shuffle equations.

**Lemma 3** There are \( \binom{n}{k} \) shuffle relations defining \( G_q(k, n) \) in the Plücker embedding.

**Example 4** \( G_q(2, 4) \) is described by a single relation:

\[
x_{12}x_{34} - x_{13}x_{24} + x_{14}x_{23} = 0.
\]

The balls of radius \( 2t \) (with respect to the subspace distance) around some \( U \in G_q(k, n) \), denoted by \( B_{2t}(U) \), can be described by explicit equations in the Plücker embedding. For it we need the Bruhat order:

\[
(i_1, \ldots, i_k) \geq (j_1, \ldots, j_k) \iff \forall l \in \{1, \ldots, k\}.
\]

Note, that the Bruhat order is not a total but only a partial order on \( \binom{n}{k} \).

**Example 5** According to the Bruhat order it holds that \( (1, 2, 7) \leq (2, 3, 7) \). But the fact that \( (2, 4, 6) \nless (2, 3, 7) \) does not imply that \( (2, 4, 6) > (2, 3, 7) \). These two tuples are not comparable.

The equations defining the balls are easily determined in the following special case:

**Proposition 6** \([10, 18]\) Define \( U_0 := \text{rs}[I_k 0_{k \times n-k}] \). Then

\[
B_{2t}(U_0) = \{ V = \text{rs}(V) \in G_q(k, n) \mid M_{i_1, \ldots, i_k}(V) = 0 \\
\forall (i_1, \ldots, i_k) \nless (t + 1, \ldots, k, n - t + 1, \ldots, n) \}\}
\]

With the knowledge of \( B_{2t}(U_0) \) we can also express \( B_{2t}(U) \) for any \( U \in G_q(k, n) \). For this note, that for any \( U \in G_q(k, n) \) there exists an \( A \in GL_n \) such that \( U_0 = A. \) Moreover,

\[
B_{2t}(U_0 A) = B_{2t}(U_0). A
\]

The following results are taken from \([18]\), where also the respective proofs can be found.

For simplifying the computations we define \( \varphi \) on \( GL_n \), where we denote by \( A[i_1, \ldots, i_k] \) the submatrix of \( A \) that consists of the rows \( i_1, \ldots, i_k \):

\[
\varphi : GL_n \longrightarrow GL(\binom{n}{k})
\]

\[
A \mapsto \begin{pmatrix}
\det A[1, \ldots, k] & \ldots & \det A[k+1, \ldots, n] \\
\vdots & \ddots & \vdots \\
\det A[n-k+1, \ldots, n] & \ldots & \det A[n-k+1, \ldots, n]
\end{pmatrix}
\]

**Lemma 7** Let \( U \in G_q(k, n) \) and \( A \in GL_n \). It holds that

\[
\varphi(U A) = \varphi(U) \varphi(A).
\]

**Theorem 8** Let \( U = U_0 A \in G_q(k, n) \). Then

\[
B_{2t}(U) = B_{2t}(U_0 A) = \{ V = \text{rs}(V) \in G_q(k, n) \mid M_{i_1, \ldots, i_k}(V) \varphi(A^{-1}) = 0 \\
\forall (i_1, \ldots, i_k) \nless (t + 1, \ldots, k, n - t + 1, \ldots, n) \}\}
\]

There are always several choices for \( A \in GL_n \) such that \( U_0 A = U. \) Since \( GL(\binom{n}{k}) \) is very large we try to choose \( A \) as simple as possible. We will now explain one such construction.

**Construction 1** For a given \( U = \text{rs}(U) \in G_q(k, n) \) we construct \( A \in GL_n \) such that \( U_0 A = U \) as follows:

1. The first \( k \) rows of \( A \) are equal to the matrix representation \( U \) of \( U. \)
2. Find the pivot columns of \( U \) (assume that \( U \) is in the reduced row echelon form).
3. Fill up the respective columns of \( A \) with zeros in the lower \( n - k \) rows.
4. Fill up the remaining submatrix of size \( n - k \times n - k \) with an identity matrix.

Then the inverse of \( A \) can be computed as follows:

1. Find a permutation \( \sigma \in S_n \) that permutes the columns of \( A \) such that
\[
\sigma(A) = \begin{pmatrix} I_k & U'' \\ 0 & I_{n-k} \end{pmatrix}.
\]
2. Then the inverse of that matrix is
\[
\sigma(A)^{-1} = \begin{pmatrix} I_k & -U'' \\ 0 & I_{n-k} \end{pmatrix}.
\]
3. Apply \( \sigma \) on the rows of \( \sigma(A)^{-1} \). The result is \( A^{-1} \). One can easily see this if one represents \( \sigma \) by a matrix \( S \). Then one gets \( (SA)^{-1}S = A^{-1}S^{-1}S = A^{-1} \).

Thus, we know how to describe the balls of a given radius \( 2t \) around an element of \( G_q(k,n) \) with linear equations in the Plücker embedding, which is exactly what is needed for a list decoding algorithm. In the following section we will describe a way of representing a subset of the Plücker coordinates of lifted rank-metric codes as linear block codes, which can then be used to come up with a list decoding algorithm in the Plücker embedding.

\section*{3 List Decoding of Lifted Gabidulin Codes in the Plücker Embedding}

\subsection*{3.1 Linear Block Codes over \( \mathbb{F}_q \) in the Plücker Coordinates of LG Codes}

Let \( C \) be an \([k \times (n-k), (n-k)(k-\delta+1, \delta)]\) Gabidulin MRD code over \( \mathbb{F}_q \). Then by Theorem 1 its lifting is a code \( \mathcal{C} \) of size \( q^{(n-k)(k-\delta+1)} \) in the Grassmannian \( G_q(k,n) \). Let
\[
x^A = [x^A_{1,\ldots,k} : \ldots : x^A_{n-k+1\ldots n}] \in \mathbb{P}(\mathcal{G})^{-1}
\]
be a vector which represents the Plücker coordinates of a subspace \( A \in G_q(k,n) \). If \( x^A \) is normalized (i.e. the first non-zero entry is equal to one), then \( x^A_{1,\ldots,k} = 1 \) for any \( A \in \mathcal{C} \).

Let \( [k] = \{1, 2, \ldots, k\} \), and let \( i = \{i_1, i_2, \ldots, i_k\} \) be a set of indices such that \(|i \cap [k]| = k - 1\). Let \( t \in k \), such that \( t > k \), and \( s = [k] \setminus i \).

**Lemma 9** Consider \( A \in \mathcal{C} \) and \( A = rs[I_k A] \). If \( x^A \) is normalized, then \( x^A_{t,k} = (-1)^{k-s}A_{s,t-k} \).

**Proof** It holds that \( x^A \) is normalized if its entries are the minors of the reduced row echelon form of \( A \), which is \([I_k A]\). Because of the identity matrix in the first \( k \) columns, the statement follows directly from the definition of the Plücker coordinates. \( \square \)

Note, that we have to worry about the normalization since \( x^A \) is projective. In the following we will always assume that any element from \( \mathbb{P}(\mathcal{G})^{-1} \) is normalized.

With Lemma 9 one can easily show, that a subset of the Plücker coordinates of a lifted Gabidulin code forms a linear code over \( \mathbb{F}_q \):

**Theorem 10** The restriction of the set of Plücker coordinates of an \((n,q^{(n-k)(k-\delta+1)},2\delta,k)\) lifted Gabidulin code \( C \) to the set \( \{i : [i] = k, |i \cap [k]| = k - 1\} \) forms a linear code \( C' \) over \( \mathbb{F}_q \) of length \( k(n-k) \), dimension \((n-k)(k-\delta+1)\) and minimum distance \( d_{\min} \geq \delta \).
Proof Since the Gabidulin code $C$ is linear, it holds that for every $A, B \in C$ we have $A + B \in C$. Together with Lemma 9 we have the same property when we consider the restriction of the set of Plücker coordinates of a lifted Gabidulin code to the set $\{ \tilde{x} : |\tilde{x}| = k, |\tilde{x} \cap [k]| = k - 1 \}$. This set is of size $k(n-k)$, and therefore we obtain a linear code $C^p$ of length $k(n-k)$ and the same dimension as $C$, i.e. $(n-k)(k-\delta+1)$. Since the rank of each non-zero $A \in C$ is greater or equal to $\delta$, also the number of non-zero entries of $A$ has to be greater or equal to $\delta$, hence the minimum Hamming distance $d_{\text{min}}$ of $C^p$ satisfies $d_{\text{min}} \geq \delta$. □

Example 11 Let $\alpha \in \mathbb{F}_{2^2}$ be a primitive element, fulfilling $\alpha^2 = \alpha + 1$. Let $C$ be the $[2 \times 2, 2, \delta = 2]$ Gabidulin MRD code over $\mathbb{F}_2$ defined by the generator matrix $G = (\alpha \ 1)$. In this example we want to consider the lifting of $C = \{(\alpha a, b) : b \in \mathbb{F}_{2^2}\}$. The codewords of $C$, their representation as $2 \times 2$ matrices, their lifting to $\mathbb{F}_q$ matrices, their lifting to $\mathbb{G}_2(2,4)$ and the respective Plücker coordinates are given in the following table.

<table>
<thead>
<tr>
<th>vector representation</th>
<th>matrix representation</th>
<th>lifting</th>
<th>Plücker coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0,0)$</td>
<td>(0 0)</td>
<td>(1 0 0 0)</td>
<td>$[1:0:0:0:0:0]$</td>
</tr>
<tr>
<td>$(\alpha,1)$</td>
<td>(0 1)</td>
<td>(1 0 1 1)</td>
<td>$[1:1:0:0:1:1]$</td>
</tr>
<tr>
<td>$(\alpha^2,\alpha)$</td>
<td>(1 0)</td>
<td>(1 0 1 1)</td>
<td>$[1:0:1:1:1:1]$</td>
</tr>
<tr>
<td>$(1,\alpha^2)$</td>
<td>(1 1)</td>
<td>(1 0 1 1)</td>
<td>$[1:1:1:1:0:1]$</td>
</tr>
</tbody>
</table>

In this example, $C^p = \{(0000), (1001), (0111), (1110)\}$. This is a $[4,2,2]$ linear code in the Hamming space. Its parity-check matrix is

$$H^p = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$ 

In other words, a Plücker coordinate vector $[x_{12} : x_{13} \ldots x_{23} : x_{24} : x_{34}]$ of a vector space from $\mathbb{G}_2(2,4)$ represents a codeword of the lifted Gabidulin code from above if and only if $x_{12} = 1$, $x_{14} + x_{23} = 0$, and $x_{13} + x_{23} + x_{24} = 0$.

3.2 The List Decoding Algorithm

We now have all the machinery needed to describe a list decoding algorithm for lifted rank-metric codes in the Plücker coordinates under the assumption that the received word has the same dimension as the codewords. Consider a lifted Gabidulin code $C \subseteq \mathbb{G}_q(k,n)$ and denote its corresponding $[k(n-k), (n-k)(k-\delta+1)]$-linear block code over $\mathbb{F}_q$ by $C^p$. The corresponding parity check matrix is denoted by $H^p$. Let $R = rs(R) \in \mathbb{G}_q(k,n)$ be the received word. Let $e$ be the number of errors (i.e. insertions and deletions) to be corrected.

We showed in Section 3.1 how a subset of the Plücker coordinates of a LG code forms a linear block code that is defined through the parity check matrix $H^p$. Since we want to describe a list decoding algorithm inside the whole set of Plücker coordinates, we define an extension of $H^p$ as follows:

$$H_{\ell}^p = \begin{pmatrix} 0_{(\delta-1)(n-k)} & H^p & 0_{(\delta-1)(n-k) \times \ell} \end{pmatrix}$$

where $\ell = \binom{n}{k} - k(n-k) - 1$. Then $[x_{1-k} : \ldots : x_{n-k+1 \ldots n}]H_{\ell}^p = 0$ gives rise to the same equations as $[x_{i_1} : \ldots : x_{i_{k(n-k)}}]H^pT = 0$, for $i_1, \ldots, i_{k(n-k)} \in \mathbb{F}$. For simplicity we will sometimes write $\tilde{x}$ for $[x_{1-k} : \ldots : x_{n-k+1 \ldots n}]$ in the following.

Theorem 12 Algorithm 1 outputs the complete list $L$ of codewords (in Plücker coordinate representation), such that for each element $\tilde{x} \in L$, $d_{\mathbb{G}}(\varphi^{-1}(\tilde{x}), R) \leq 2e$.
Algorithm 1

Input: \( \mathcal{R}, c \)

1. Find the (linear) equations defining \( B_{2e}(\mathcal{R}) \) in the Plücker coordinates, as explained in Section 2.2.
2. Solve the system of (linear) equations, that arises from \( \bar{x}Hp = 0 \), together with the equations of \( B_{2e}(\mathcal{R}) \), the (bilinear) shuffle relations and the equation \( x_{1,\ldots,k} = 1 \).

Output: The solutions \( \bar{x} = [x_{1\ldots k} : \ldots : x_{n-k+1\ldots n}] \) of this system of equations.

Proof The solution set to the shuffle relations is exactly \( \varphi(\mathcal{G}_q(k,n)) \), i.e. all the elements of \( \mathbb{P}^\binom{n}{k} \) that are Plücker coordinates of a \( k \)-dimensional vector space in \( \mathbb{F}_q^n \). The subset of this set with the condition \( x_{1,\ldots,k} = 1 \) is exactly the set of Plücker coordinates of elements in \( \mathcal{G}_q(k,n) \) whose reduced row echelon form has \( I_k \) as the left-most columns. Intersecting this with the solution set of the equations given by \( H^p \) achieves the Plücker coordinates of the lifted code \( \mathcal{C} \). The intersection with \( B_{2e}(\mathcal{R}) \) is then given by the additional equations from Step 1 in the algorithm. Thus the solution set to the whole system of equation is the Plücker equations of \( \mathcal{C} \cap B_{2e}(\mathcal{R}) \).

For the analysis of complexity of this algorithm we need to calculate the number of equations, denoted by \( \tau \), that define a ball of radius \( 2e \).

Lemma 13 The number of equations defining \( B_{2e}(\mathcal{U}_0) \) is equal to the number of equations defining \( B_{2e}(\mathcal{U}) \) for any \( \mathcal{U} \in \mathcal{G}_q(k,n) \).

Proof Follows directly from Lemma 7.

Since we can count the elements that are not less than or equal to a given element in the Bruhat order, we get:

Lemma 14 The number of equations defining \( B_{2e}(\mathcal{U}) \) inside \( \mathcal{G}_q(k,n) \) is

\[
\tau = \sum_{l=0}^{k-e} \binom{n-k}{k-l} \binom{k}{l} = \sum_{l=k-e}^{k} \binom{n-k}{k-l} \binom{k}{l}.
\]

Proof The condition that \((i_1,\ldots,i_k) \not\leq (e+1,\ldots,k,n-l+1,\ldots,n)\) is equivalent to

\[\exists l \in \{1,\ldots,k-e\} : i_l > k.\]

For such an \( l \) there are \( k - l + 1 \) entries chosen freely from \( \{k+1,\ldots,n\} \) and \( l-1 \) entries from \( \{1,\ldots,k\} \). Hence there are

\[
\sum_{l=1}^{k-e} \binom{n-k}{k-l+1} \binom{k}{l-1} = \sum_{l=0}^{k-e-1} \binom{n-k}{k-l} \binom{k}{l}
\]

many elements in \( \binom{n}{k} \) that are \( \not\leq (e+1,\ldots,k,n-l+1,\ldots,n) \), which is equal to the number of equations defining \( B_{2e}(\mathcal{U}) \).

The complexity of Algorithm 1 is dominated by solving the system of \( \tau + 1 + (d-1)(n-k) + \binom{n}{2k} \) linear and bilinear equations in \( \binom{n}{k} \) variables. This has a complexity that is polynomial in \( n \) and exponential in \( k \).

In most of the examples we computed though, we only needed a subset of all equations to get the solutions. For this note, that the actual information is encoded in the rank-metric code part of the matrix representation of the vector space, i.e. in the Plücker coordinates corresponding to \( CP \). Hence, one does not need the \( k \times n \)-matrix representation of the solutions from an application point of view, since the information can be extracted directly from the Plücker coordinate representation of the vector spaces. On the other hand, because of this structure it is also straight-forward to construct the matrix representation by using Lemma 9 (i.e. without any computation needed). So, the number of variables in the system could be reduced to \( k(n-k) \), and this can decrease the complexity of the algorithm.
Example 15 We consider the $(4,4,4,2)$ lifted Gabidulin code from Example 11. Note, that for a received space of dimension 2 it is not possible to decode to a unique closest codeword.

1. Assume we received
   \[ R_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \]
   We would like to correct one error. Thus we first find the equations for the ball of subspace radius 2:
   \[ B_2(\mathcal{U}_0) = \{ V = rs(V) \in \mathcal{G}_2(2,4) \mid M_{3,4}(V) = 0 \} \]
   We construct $A_1^{-1}$ according to Construction 1
   \[ A_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \]
   and compute the last column of $\varphi(A_1^{-1})$:
   \[ [1 : 0 : 0 : 1 : 0 : 0]^T. \]
   Thus, we get that
   \[ B_2(R_1) = \{ V = rs(V) \in \mathcal{G}_2(2,4) \mid M_{1,4}(V) + M_{2,3}(V) = 0 \}. \]
   Then combining with the parity check equations from Example 11 we obtain the following system of linear equations to solve
   \[
   \begin{align*}
   x_{13} + x_{14} + x_{24} &= 0 \\
   x_{14} + x_{23} &= 0 \\
   x_{12} + x_{23} &= 0 \\
   x_{12} &= 1
   \end{align*}
   \]
   where the first two equations arise from $\bar{H}^p$, the third from $B_2(R_1)$ and the last one represents the identity submatrix. This system has the two solutions $(1,1,1,0,1,0)$ and $(1,0,1,1,0,1)$ for $(x_{12},x_{13},x_{14},x_{23},x_{24})$. Since we used all the equations defining the ball in the system of equations, we know that the two codewords corresponding to these two solutions (i.e. the third and fourth in Example 11) are the ones with distance 2 from the received space, and we do not have to solve $x_{34}$ at all. The corresponding codewords are
   \[
   \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.
   \]

2. Now assume we received
   \[ R_2 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}. \]
   As previously, we construct $A_2^{-1}$ according to Construction 1
   \[ A_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \]
   and compute the last column of $\varphi(A_2^{-1})$:
   \[ [1 : 1 : 0 : 1 : 1 : 1]^T. \]
Thus, we get that
\[ B_2(R_1) = \{ V = rs(V) \in G_2(2, 4) \mid M_{1,2}(V) + M_{1,3}(V) + M_{2,3}(V) + M_{2,4}(V) + M_{3,4}(V) = 0 \}. \]

Then combining with the parity check equations from Example 11 and the shuffle relation from Example 4 we obtain the following system of linear and bilinear equations:
\[
\begin{align*}
&x_{13} + x_{14} + x_{24} = 0 \\
x_{14} + x_{23} = 0 \\
x_{12} + x_{13} + x_{23} + x_{24} + x_{34} = 0 \\
x_{12}x_{34} + x_{13}x_{24} + x_{14}x_{23} = 0 \\
x_{12} = 1
\end{align*}
\]

We rewrite these equations in terms of the variables \( x_{13}, x_{14}, x_{23}, x_{24} \) which correspond to a lifted Gabidulin code as follows.
\[
\begin{align*}
x_{13} + x_{14} + x_{24} &= 0 \\
x_{14} + x_{23} &= 0 \\
x_{13} + x_{23} + x_{24} + x_{13}x_{24} + x_{14}x_{23} &= 1
\end{align*}
\]

This system has three solutions \((1, 0, 0, 1), (0, 1, 1, 1), \) and \((1, 1, 1, 0)\) for \((x_{13}, x_{14}, x_{23}, x_{24})\). The corresponding codewords are
\[
\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.
\]

**Remark 16** Note that an upper bound for the list size, i.e. the number of codewords in a ball of subspace radius \(2\epsilon\) around a received word, can be directly derived from the bound on a list size of a classical Gabidulin code, given rank radius \(e\). This result follows from the next theorem.

**Theorem 17** The list size of a list decoder for lifted Gabidulin codes is less than or equal the list size of a list decoder for classical Gabidulin codes (with the corresponding parameters), with equality when the received word is of the type \( R = rs[ I_k A ] \in G_q(k, n) \) for some \( A \in F_q^{k \times (n-k)} \).

**Proof** Assume \( R = rs[ I_k A ] \in G_q(k, n) \). Then \( d_S(R, rs[ I_k B ]) = 2\text{rank}(A - B) \), for any \( B \in F_q^{k \times (n-k)} \), hence the list size is equal.

For the general case, i.e. when \( R \) has arbitrary shape, we prove that the list size \(|B_{2\epsilon}(R)| \leq |B_{2\epsilon}(rs[ I_k \tilde{R} ])|\) for some \( \tilde{R} \in F_q^{k \times (n-k)} \). For this let \( R \) be the reduced row echelon form of \( R \). We can write \( R \) as follows
\[
R = \begin{pmatrix} J_{\ell \times k} & R_1 \\ 0_{(k-\ell) \times k} & R_2 \end{pmatrix},
\]
such that \( \text{rank}(J_{\ell \times k}) = \ell \). If some codeword \( U = rs[ I_k B ] \) of the lifted Gabidulin code is in the ball \( B_{2\epsilon}(R) \), then \( \dim(U \cap R) \geq k - e \) and this intersection can only happen in the row space of \( (J_{\ell \times k} \bar{R}_1) \). It follows that \( U \) is also in the ball of radius \(2\epsilon\) around the row space of
\[
R^* = \begin{pmatrix} J_{\ell \times k} & R_1 \\ J_C & R_2 \end{pmatrix},
\]
where \( J_C \in F_q^{(k-\ell) \times k} \) such that the first \( k \) columns of \( R^* \) form a matrix of rank \( k \). It holds that \( rs(R^*) = rs[ I_k \tilde{R} ] \) for some \( \tilde{R} \in F_q^{k \times (n-k)} \) and thus the statement follows.

Bounds for the list size for classical Gabidulin list decoding can be found in [26].
Conclusion and Open Problems

We presented a list decoding algorithm for lifted Gabidulin codes that works by solving a system of linear and bilinear equations in the Plücker coordinates. In contrast to the algorithms presented in [9,15] this algorithm works for lifted Gabidulin codes for any set of parameters $q, n, k, \delta$.

One can easily extend the algorithm presented in this paper to work also for received spaces of a different dimension. For this, one only needs to change the conditions in Proposition 6 indicating which Plücker coordinates have to be zero. The rest of the theory can then be carried over straightforwardly. In a similar manner one can make the algorithm work for unions of LG codes of different length (cf. e.g. [21]). To do so, one needs to add a preliminary step in the algorithm where a rank argument decides, which of these LG codes can possibly have codewords that are in the ball around the received word.

The storage needed for our algorithm is fairly little, the complexity is polynomial in $n$ but exponential in $k$. Since in applications, $k$ is quite small while $n$ tends to get large, this is still reasonable. In future work, we still want to improve this complexity by trying to decrease the size of the system of equations to solve in the last step of the algorithm. Moreover, it would be interesting to see if some converse version of Theorem 10 exists, i.e. if one can generate constant dimension codes from a given linear block code by using this as a subset of the Plücker coordinates of the constant dimension code. Moreover, we would like to find other families of codes that can be described through equations in their Plücker coordinates and use this fact to come up with list decoding algorithms of these other codes.

References

On the Existence of $q$-Analogs of Steiner Systems

Michael Braun · Tuvi Etzion · Patric R. J. Östergård · Alexander Vardy · Alfred Wassermann

Received: date / Accepted: date

Abstract A $q$-analog of a Steiner system (briefly, $q$-Steiner system), denoted by $S = S_q[t,k,n]$, is a set of $k$-dimensional subspaces of $F_q^n$ such that each $t$-dimensional subspace of $F_q^n$ is contained in exactly one element of $S$. Presently, $q$-Steiner systems are known only for $t = 1$ and in the trivial cases $t = k$ and $k = n$. In this paper, the first known nontrivial $q$-Steiner systems with $t \geq 2$ are constructed. Specifically, $S_2[2,3,13]$ $q$-Steiner systems are found by requiring that their automorphism group contain the normalizer of a Singer subgroup of $GL(13,2)$. This approach leads to an instance of the exact cover problem, which turns out to have many solutions.

Keywords exact cover · Frobenius automorphism group · $q$-analog · Singer subgroup · spread · Steiner system

T. Etzion was supported in part by the Israeli Science Foundation (ISF), Jerusalem, Israel, under Grant 10/12. P.R.J. Östergård was supported in part by the Academy of Finland, under Grant No. 132122.

Michael Braun
Faculty of Computer Science
University of Applied Sciences Darmstadt
D-64295 Darmstadt, Germany

Tuvi Etzion
Department of Computer Science, Technion
Haifa 32000, Israel

Patric R. J. Östergård
Department of Communications and Networking
Aalto University School of Electrical Engineering
P.O. Box 13000, FI-00076 Aalto, Finland

Alexander Vardy
Department of Electrical and Computer Engineering
University of California San Diego
La Jolla, CA 92093, USA

Alfred Wassermann
Department of Mathematics
University of Bayreuth
D-95440 Bayreuth, Germany
1 Introduction

An $S(t, k, v)$ Steiner system is a collection of $k$-subsets, called blocks, of a $v$-set (of points), such that each $t$-subset of the $v$-set is contained in exactly one block. This paper is devoted to the existence of $q$-analogs of Steiner systems, also known as $q$-Steiner systems. Throughout this work, we assume that $q \geq 2$ is a prime power.

An $S_q[t, k, n]$ $q$-Steiner system is a collection $S$ of $k$-dimensional subspaces of $\mathbb{F}_q^n$ ($k$-subspaces for short) such that each $t$-subspace of $\mathbb{F}_q^n$ is contained in exactly one element of $S$. Such $q$-Steiner systems can be readily constructed for $t = k$ and $k = n$ and are called trivial. Moreover, $S_q[1, k, n]$ $q$-Steiner systems are known as spreads and exist if and only if $k$ divides $n$ [21, Chap. 24]. No other $q$-Steiner systems are known so far.

Indeed, the existence of nontrivial $S_q[t, k, n]$ $q$-Steiner systems with $t \geq 2$ has tantalized researchers for decades and has been addressed in various studies including [2,9,22,24,27,28]. In particular, much attention has been put on the smallest possible case of $S_q[2,3,7]$. Actually, Metsch [22] conjectured that nontrivial $q$-Steiner systems for $t \geq 2$ do not exist. Whereas positive results have been missing for $q$-Steiner systems, there has been progress on the more general problem of finding $q$-analogs of $t$-designs, for which it is required that each $t$-subspace be contained in $\lambda$ $k$-subspaces [5,23,25–28]. Also the corresponding packing and covering problems, with the requirement that each $t$-subspace be contained in at most one and at least one $k$-subspace, respectively, have been addressed [7–9,19].

It should be emphasized that one of the main reasons why these issues have received a lot of attention in the recent years are their applications for error-correction in networks, under randomized network coding, as shown in [8,18].

In the current paper, we carry out a computer-aided search for $S_q[2,3,13]$ $q$-Steiner systems after imposing a structure on these via a prescribed group of automorphisms. Indeed, this approach is successful and leads to the discovery of the first known nontrivial $q$-Steiner system with $t \geq 2$.

The paper is organized as follows. In Section 2 we consider automorphisms of Steiner structures in general and the normalizer of a Singer subgroup of a general linear group in particular. The computational problem that arises in the current work and a solution are presented in Section 3, which is concluded by a discussion of some corollaries of the new result and problems for further research.

2 Automorphisms of $q$-Steiner Systems

We consider the action of the group $G$ of bijective incidence-preserving mappings on the set of subspaces of $\mathbb{F}_q^n$. We know from the Fundamental Theorem of Projective Geometry [3] that $G \cong \Gamma L(n, q)$ (the general semilinear group), which in turn is isomorphic to the general linear group $GL(n, q)$ when $q$ is a prime.

The action of $G \cong \Gamma L(n, q)$ on single subspaces extends in the obvious way to sets of subspaces, and thereby to $q$-Steiner systems. Given a set $S$ of subspaces of $\mathbb{F}_q^n$ and a group element $g \in G$, we denote the image of $S$ under the action of $g$ by $S^g$. We say that two sets of subspaces $S_1$ and $S_2$ are isomorphic if there exists an element $g \in G$ such that $S_2 = S_1^g$. An element $g \in G$ for which $S^g = S$ is called an automorphism of $S$. The automorphisms of a set $S$ of subspaces form a group under composition, called the (full) automorphism group and denoted $\text{Aut}(S)$. A
A subgroup of $\text{Aut}(S)$ is called a group of automorphisms. As $G$ acts transitively on the set of $k$-subspaces for any fixed $k$, the automorphism group of a nontrivial $q$-Steiner system is necessarily a proper subgroup of $G$.

Throughout the rest of this work, we assume that $q$ is prime, whereby $G \cong \text{GL}(n, q)$.

A standard technique for finding combinatorial objects is to prescribe a group of automorphisms and search for corresponding objects. For surveys of the theory and applications in design theory, where this approach has been very successful along the years, the reader is referred to [10, Sect. 6.3] and [15, Sect. 9.2]. By prescribing a group of automorphisms, the construction problem is simplified, but choosing the right groups can be a challenge. We shall now discuss certain apposite subgroups of $\text{GL}(n, q)$.

Vectors in $\mathbb{F}^n_q$ can be interpreted as elements of the field $\mathbb{F}_{q^n}$. Let $\alpha$ be a primitive element in $\mathbb{F}_{q^n}$. A Singer cycle of $\text{GL}(n, q)$ is an element of order $q^n - 1$. Such elements always exist, and correspond to multiplication by $\alpha$ in the multiplicative group $\mathbb{F}^*_q = \{\alpha^0, \alpha^1, \ldots, \alpha^{q^n-2}\}$ of $\mathbb{F}_{q^n}$. The group generated by a Singer cycle is called a Singer subgroup and its $q^n - 1$ elements correspond to multiplication by $\alpha^i$, $0 \leq i \leq q^n - 2$ in $\mathbb{F}^*_q$.

Another subgroup of $\text{GL}(n, q)$ is obtained via the Frobenius automorphism group of $\mathbb{F}_{q^n}$, which is cyclic of order $n$ and maps $\alpha^i$ to $(\alpha^i)^q$ in $\mathbb{F}^*_q$, where $i$, $0 \leq i \leq n - 1$ specifies an element in the group. The normalizer of a subgroup $H \leq G$ is defined as $N_G(H) = \{g \in G : gHg^{-1} = H\}$. The following well-known result can be found in, for example, [13, pp. 187, 188].

**Theorem 1** The normalizer of a Singer subgroup of $\text{GL}(n, q)$ has order $n(q^n - 1)$ and is isomorphic to the semidirect product of the Singer subgroup and the Frobenius automorphism group of $\mathbb{F}_{q^n}$.

The following theorem follows from a more general result by Kantor [14] and is stated explicitly in [6].

**Theorem 2** Let $n$ be a prime. Then, except when $n = q = 2$, the normalizer of a Singer subgroup is a maximal subgroup of $\text{GL}(n, q)$.

A Singer subgroup and the normalizer of a Singer subgroup have earlier been successful choices when prescribing automorphisms for various types of $q$-analog structures [5, 8]. The following lemma for the case $q = 2$, $t = 2$ is useful in the sequel.

**Lemma 1** A collection $S$ of $k$-subspaces of $\mathbb{F}^n_2$ is an $S_2[2, k, n]$ $q$-Steiner system if and only if each pair of distinct non-zero vectors from $\mathbb{F}^n_2$ occurs in exactly one element of $S$.

**Proof** This follows from the definition of a $q$-Steiner system and the fact that any two of the three non-zero vectors in a 2-subspace, $u$ and $v$, defines the third one, $u + v$. \( \square \)

We shall now connect an $S_2[2, k, n]$ with a Singer subgroup as a group of automorphisms to a difference family. A $(v, k, \lambda)$ difference family over an additive
group $G$ is a collection $B_1, B_2, \ldots, B_m$ of $k$-subsets of $G$ such that every nonidentity element of $G$ occurs $\lambda$ times in the multiset $\{a - b : a, b \in B_i, \ a \neq b, 1 \leq i \leq m\}$.

For a set $X = \{x_1, x_2, \ldots, x_m\}$ of powers of a primitive element $\alpha$, such as the non-zero elements of a subspace, we define $\text{inv}(X) := \{x_i - x_j : i \neq j\}$.

**Lemma 2** If two sets $X, Y$ of powers of $\alpha \in \mathbb{F}_2^n$ are in the same orbit under the Singer cycle, then $\text{inv}(X) = \text{inv}(Y)$.

**Theorem 3** If there exists an $S_2[2, k, n]$-Steiner system with the Singer subgroup as a group of automorphisms, then there exists a $(2^n - 1, 2^k - 1, 1)$ difference family over the group $\mathbb{Z}_{2^n - 1}$.

**Proof** We show that a transformation between the involved structures via the mapping between vectors in $\mathbb{F}_2^n$ and powers of a primitive element $\alpha$ in the multiplicative group $\mathbb{F}_2^\ast$ gives the desired result.

Consider an $S_2[2, k, n]$-Steiner system $S$ with a Singer subgroup as a group of automorphisms, and carry out the above mentioned transformation to the nonzero vectors in the orbit representatives of the $k$-dimensional elements of $S$. In this manner, $(2^k - 1)$-subsets of $\mathbb{F}_2^n$ are obtained and, by extracting the power $i$ of each element $\alpha^i$, $(2^k - 1)$-subsets of $\mathbb{Z}_{2^n - 1}$.

Now consider an arbitrary nonidentity element $a \in \mathbb{Z}_{2^n - 1}$. This element can be obtained as a difference in exactly $2^n - 1$ ways: $(a + i) - i$ for $0 \leq i \leq 2^n - 2$. By Lemma 1 each pair $\{a + i, i\}$ occurs in exactly one element of $S$ and by Lemma 2 such a pair occurs in an orbit representative. Consequently, the difference $a$ occurs once in exactly one of the $(2^k - 1)$-subsets of $\mathbb{Z}_{2^n - 1}$.

The implication in Theorem 3 also works in the opposite direction if one restricts the difference families considered to those that are mapped to subspaces with the transformation used. Indeed, theoretical results needed in the current work can be adopted from the context of difference families, developed and discussed in, for example, [1] and [4, Chap. VII].

A generator of the Frobenius automorphism group corresponds to a multiplier in the framework of difference families. We cannot get arbitrary multipliers as only elements of $\text{GL}(n, 2)$ are considered. For elements of the Frobenius automorphism group of $\mathbb{F}_{q^n}$ we do know that $a + b = c$ implies that $a^{q^i} + b^{q^i} = (a + b)^{q^i} = c^{q^i}$.

### 3 Computer Search and Results

The problem of finding a $q$-analogue of a $t$-design can be formulated in terms of a system of linear Diophantine equations. Let $M$ be a 0-1 matrix with rows and columns corresponding to the $t$-subspaces and the $k$-subspaces of $\mathbb{F}_q^n$, respectively. We assume that $k > t$ and let the 1s in $M$ denote inclusion. By definition, a solution to $Mx = [\lambda, \lambda, \ldots, \lambda]^T$ is then a $q$-analogue of a $t$-design. The problem of finding solutions for these large systems is commonly out of computational feasibility for interesting parameters, but the problem of finding $q$-analogs of a $t$-design with a prescribed group $A$ can also be handled within this framework. With prescribed automorphisms, the entries of the matrix $M^A$, where rows and columns correspond to $A$-orbits instead of single subspaces, may be larger than 1.
This technique—which is analogous to a standard technique in design theory that is called the Kramer–Mesner method after its developers—is described in detail in [5].

For the case of $\lambda = 1$, that is, for $S_q[t,k,n]$ $q$-Steiner systems, the system of linear Diophantine equations can be reduced to an instance of the exact cover problem [17]. For algorithms and software for solving instances of the exact cover problem, see [16,17].

For $S_2[2,3,13]$ $q$-Steiner systems we get a matrix $M$ with $11\,180\,715$ rows and $3\,269\,560\,515$ columns. Since an $S_2[2,3,13]$ $q$-Steiner system consists of $1\,597\,245$ 3-subspaces, a solution vector has to contain that many 1s. It is apparent that it is not feasible to attack the corresponding instance in a direct way with existing algorithms. However, by prescribing a group along the discussion in the previous section, one can reduce the size of the instance.

By prescribing the normalizer of a Singer subgroup of $GL(13,2)$ as a group $A$ of automorphisms, we arrive at an instance of the exact cover problem represented by the $A$-incidence matrix $M^A$ with $105$ rows and $30\,705$ columns of which $25\,572$ columns have admissible entries less than 2. Since $n = 13$ is a prime, all orbits represented by the columns of $M^A$ have trivial stabilizers in $A$ and hence full length $|A| = 106\,483$. Consequently, we want to find $1\,597\,245/106\,483 = 15$ columns in $M^A$ such that each row is covered exactly by one column.

In our implementation, we considered the primitive element $\alpha \in F_{2^{13}}$ that is a root of the primitive polynomial $x^{13} + x^4 + x^3 + x + 1$. A computer search then found several solutions for this instance of the exact cover problem, one of which is as follows (the powers of $\alpha$ of the nonzero elements in the 3-subspaces are listed):

\[
\begin{align*}
\{0,1,1249,5040,7258,7978,8105\}, & \quad \{0,7,1857,6681,7259,7381,7908\}, \\
\{0,9,1144,1945,6771,7714,8102\}, & \quad \{0,11,209,1941,2926,3565,6579\}, \\
\{0,12,2181,2519,3696,6673,6965\}, & \quad \{0,13,4821,5178,7823,8052,8110\}, \\
\{0,17,291,1199,5132,6266,8057\}, & \quad \{0,20,1075,3939,3996,4776,7313\}, \\
\{0,21,2900,4226,4915,6087,8008\}, & \quad \{0,27,1190,3572,4989,5199,6710\}, \\
\{0,30,141,682,2024,6256,6406\}, & \quad \{0,31,814,1161,1243,4434,6254\}, \\
\{0,37,258,2993,4703,5396,6469\}, & \quad \{0,115,949,1272,1580,4539,4873\}, \\
\{0,119,490,5941,6670,6812,7312\}. 
\end{align*}
\]

On a contemporary personal computer, we found the first solution after a couple of hours. We let the program run for more than a month and during this period 401 solutions were found. However, it does not seem feasible to find all solutions of this type since estimates indicate a total run time far outside what is currently feasible, beyond one million years.

By Theorem 2, we know that the prescribed group is the (full) automorphism group of the $q$-Steiner systems in this case. Otherwise, if the full automorphism group would be $GL(n,2)$, all $k$-subspaces would be contained in one single orbit and the $q$-Steiner system would be trivial.

The fact that the group is a maximal subgroup of $GL(13,2)$ makes it possible to say much more. Actually, one can prove that the systems obtained are pairwise nonisomorphic. The following theorem and its proof are analogous to those in [20, Lemma 2.1].
Lemma 3 Let \( A \leq \text{GL}(n, q) = G \) be the automorphism group of two \( S_q[t, k, n] \) \( q \)-Steiner systems \( S_1 \) and \( S_2 \). Then an element in \( G \) that maps the \( k \)-subspaces of \( S_1 \) onto those of \( S_2 \) lies in the normalizer \( N_G(A) \).

Proof Let \( S_1^g = S_2 \) for some \( g \in G \), and consider an arbitrary \( a \in \text{Aut}(S_1) \). Since \( S_2^{g^{-1}a} = S_1^g = S_2 \), we have that \( g^{-1}ag \in \text{Aut}(S_2) = \text{Aut}(S_1) = A \). By definition, \( g \in N_G(A) \). □

Theorem 4 Let \( n \) be a prime, and let \( A \leq \text{GL}(n, q) = G \) be the normalizer of a Singer subgroup. Then two distinct nontrivial \( S_q[t, k, n] \) \( q \)-Steiner systems \( S_1 \) and \( S_2 \) with automorphism group \( A \) are nonisomorphic.

Proof Nontrivial \( q \)-Steiner systems must obviously have \( n \geq 3 \). Assume that \( S_1 \) and \( S_2 \) are isomorphic. Then, by Lemma 3, \( S_1^g = S_2 \) for some \( g \in N_G(A) \). As \( A \) is the normalizer of a Singer subgroup and \( n \) is a prime, we know by Theorem 2 that the only overgroups of \( A \)—and possible values of \( N_G(A) \)—are \( A \) and \( G \). But any normal subgroup of \( G \) is either a subgroup of the center \( Z(\text{GL}(n, q)) \) or an overgroup of \( \text{SL}(n, q) \) and \( q - 1 = |Z(\text{GL}(n, q))| < |A| < |\text{SL}(n, q)| \) when \( n \geq 3 \). Hence \( N_G(A) = A \), and \( g \in A = \text{Aut}(S_1) \) implies that \( S_1^g = S_1 \), which gives \( S_1 = S_2 \) and a contradiction. □

Obviously, there are other groups than the group considered here for which \( S_2[2, 3, 13] \) \( q \)-Steiner systems might exist. Inspired by the positive results, further attempts were made to find other \( q \)-Steiner systems, considering various parameters and groups. The following nonexistence results were obtained in this manner (extending work in [8,19]):

\[ \begin{align*}
S_2[2, 3, 7] &: \text{Frobenius automorphism group, order 7} \\
S_2[2, 3, 8] &: \text{Singer subgroup, order 255} \\
S_2[2, 4, 10] &: \text{Normalizer of Singer subgroup, order 10230} \\
S_2[2, 4, 13] &: \text{Normalizer of Singer subgroup, order 106483} \\
S_2[3, 4, 10] &: \text{Normalizer of Singer subgroup, order 10230} \\
S_3[2, 3, 7] &: \text{Singer subgroup, order 2186} \\
S_5[2, 3, 7] &: \text{Normalizer of Singer subgroup, order 546868}
\end{align*} \]

To get an \( (8191, 7, 1) \) difference family from (1), the 15 sets should be developed by, for each set \( \{s_1, s_2, \ldots, s_7\} \), constructing the sets \( \{2^is_1, 2^is_2, \ldots, 2^is_7\} \) modulo 8191 for \( 0 \leq i \leq 12 \). This gives \( 15 \cdot 13 = 195 \) sets in total. It is already known that an \( (8191, 7, 1) \) difference family exists; such a difference family is obtained in [11] using a construction developed by Wilson [20].

We get an \( S(2, 7, 8191) \) Steiner system by developing each of the 195 sets of our \( (8191, 7, 1) \) difference family by, for each set \( \{s_1, s_2, \ldots, s_7\} \), constructing the sets \( \{s_1 + i, s_2 + i, \ldots, s_7 + i\} \) modulo 8191 for \( 0 \leq i \leq 8190 \); cf. [1, Remarks 16.5].

An \( S(2, 7, 8191) \) Steiner system can also be obtained as a derived design of an \( S(3, 8, 8192) \) Steiner system, which exists by the following theorem [9].

Theorem 5 If there exists an \( S_2[2, k, n] \) \( q \)-Steiner system, then there exists an \( S(3, 2^k, 2^n) \) Steiner system.
Such an $S(3,8,8192)$ Steiner system can further be applied to various constructions, such as those in [12], to get $S(3,8,n)$ Steiner systems also for other values of $n$.

We give one further application of $q$-Steiner systems: it is proved in [2] that $q$-Steiner systems are diameter perfect codes.

There is no reason to believe that $q$-Steiner systems would not exist for any other parameters than those mentioned in the current work. We conjecture that $S_2[2,3,n]$ $q$-Steiner systems exist at least whenever $n \equiv 1 \pmod 6$ and $n \geq 13$ is a prime.

As a matter of fact, the main question is no longer whether further $q$-Steiner systems exist but how these can be found. Not only should computational attempts be carried out, but one should also consider the possibility of algebraic and combinatorial constructions of particular systems and recursively of infinite families (for example, in the framework of difference families).

The large number of isomorphism classes of $S_2[2,3,13]$ $q$-analog Steiner systems indicates that an $S_2[3,4,14]$ might exist. A general open question is whether non-trivial $S_q[t,k,n]$ $q$-analog Steiner systems exist for other parameters than $q = 2$, $t = 2$, and $k = 3$.

Acknowledgements The authors are grateful to Elisa Gorla, Joachim Rosenthal, and Amin Shokrollahi for organizing the conference “Trends in Coding Theory” in Ascona, Switzerland between October 28 and November 2, 2012, where the final pieces of the current work were put together. The authors also thank Prof. Donald Knuth for making available his dancing links software, which finally produced the very first solution.

References

Multi-Trial Guruswami–Sudan Decoding for Generalised Reed–Solomon Codes

Johan S. R. Nielsen · Alexander Zeh

February 12, 2013

Abstract An iterated refinement procedure for the Guruswami–Sudan list decoding algorithm for Generalised Reed–Solomon codes based on Alekhnovich’s module minimisation is proposed. The method is parametrisable and allows variants of the usual list decoding approach. In particular, finding the list of closest codewords within an intermediate radius can be performed with improved average-case complexity while retaining the worst-case complexity.

Keywords Guruswami–Sudan · List Decoding · Reed–Solomon Codes · Multi-Trial

1 Introduction

Since the discovery of a polynomial-time hard-decision list decoder for Generalised Reed–Solomon (GRS) codes by Guruswami and Sudan (GS) [12, 7] in the late 1990s, much work has been done to speed up the two main parts of the algorithm: interpolation and root-finding. Notably, for interpolation Beelen and Brander [2] mixed the module reduction approach by Lee and O’Sullivan [8] with the parametrisation of Zeh et al. [13], and employed the fast module reduction algorithm by Alekhnovich [1]. Bernstein [4] pointed out that a slightly faster variant can be achieved by using the reduction algorithm by Giorgi et al. [6].

For the root-finding step, one can employ the method of Roth and Ruckenstein [11] in a divide-and-conquer fashion, as described by Alekhnovich [1]. This step then becomes an order of magnitude faster than interpolation, leaving the latter as the main target for further optimisations.

For a given GRS code, the GS algorithm has two parameters, both positive integers: the interpolation multiplicity \( s \) and the list size \( \ell \). Together with the code parameters they determine the decoding radius \( \tau \). To achieve a higher decoding radii for some

J. S. R. Nielsen is with the Institute of Mathematics, Technical University of Denmark
E-mail: j.s.r.nielsen@mat.dtu.dk

A. Zeh is with the Institute of Communications Engineering, University of Ulm, Germany and Research Center INRIA Saclay - Île-de-France, École Polytechnique, France
E-mail: alex@codingtheory.eu
given GRS code, one needs higher \( s \) and \( \ell \), and the value of these strongly influence the running time of the algorithm.

In this work, we present a novel iterative method: we first solve the interpolation problem for \( s = \ell = 1 \) and then iteratively refine this solution for increasing \( s \) and \( \ell \). In each step of our algorithm, we obtain a valid solution to the interpolation problem for these intermediate parameters. The method builds upon that of Beelen–Brander \cite{2} and has the same asymptotic complexity.

The method therefore allows a fast multi-trial list decoder when our aim is just to find the list of codewords with minimal distance to the received word. At any time during the refinement process, we will have an interpolation polynomial for intermediate parameters \( \hat{s} \leq s \), \( \hat{\ell} \leq \ell \) yielding an intermediate decoding radius \( \hat{\tau} \leq \tau \). If we perform the root-finding step of the GS algorithm on this, all codewords with distance at most \( \hat{\tau} \) from the received are returned; if there are any such words, we break computation and return those; otherwise we continue the refinement. We can choose any number of these trials, e.g. for each possible intermediate decoding radius between half the minimum distance and the target \( \tau \).

Since the root-finding step of GS is cheaper than the interpolation step, this multi-trial decoder will have the same asymptotic worst-case complexity as the usual GS using the Beelen–Brander interpolation; however, the average-case complexity is better since fewer errors are more probable.

This contribution is structured as follows. In the next section we give necessary preliminaries and state the GS interpolation problem for decoding GRS codes. In Section 3 we give a definition and properties of minimal matrices. Alekhnovich’s algorithm can bring matrices to this form, and we give a more fine-grained bound on its asymptotic complexity. Our new iterative procedure is explained in detail in Section 4.

2 Preliminaries

2.1 Notation

Let \( \mathbb{F}_q \) be the finite field of order \( q \) and let \( \mathbb{F}_q[X] \) be the polynomial ring over \( \mathbb{F}_q \) with indeterminate \( X \). Let \( \mathbb{F}_q[X,Y] \) denote the polynomial ring in the variables \( X \) and \( Y \) and let \( \operatorname{wdeg}_{u,v} X^iY^j \triangleq ui + vj \) be the \((u,v)\)-weighted degree of \( X^iY^j \).

A vector of length \( n \) is denoted by \( v = (v_0,...,v_{n-1}) \). If \( v \) is a vector over \( \mathbb{F}_q[X] \), let \( \deg v \triangleq \max_i \{\deg v_i(X)\} \). We introduce the leading position as \( \text{LP}(v) = \max_i \{i | \deg v_i(X) = \deg v\} \) and the leading term \( \text{LT}(v) = v_{\text{LP}(v)} \) is the term at this position. An \( m \times n \) matrix is denoted by \( V = \|v_{i,j}\|_{i=0}^{m-1},j=0}^{n-1} \). The rows of such a matrix will be denoted by lower-case letters, e.g. \( v_0,...,v_{m-1} \). Furthermore, let \( \deg V = \sum_{i=0}^{m-1} \deg v_i \). Modules are denoted by capital letters such as \( M \).

2.2 Interpolation-Based Decoding of GRS Codes

Let \( \alpha_0,...,\alpha_{n-1} \) be \( n \) nonzero distinct elements of \( \mathbb{F}_q \) with \( n < q \) and let \( w_0,...,w_{n-1} \) be \( n \) (not necessarily distinct) nonzero elements of \( \mathbb{F}_q \). A GRS code \( \mathcal{GRS}(n,k) \) of length \( n \) and dimension \( k \) over \( \mathbb{F}_q \) is given by

\[
\mathcal{GRS}(n,k) \triangleq \{w_0 f(\alpha_0),...,w_{n-1} f(\alpha_{n-1}) : f(X) \in F_q[X], \deg f(X) < k\}. \quad (1)
\]
GRS codes are Maximum Distance Separable (MDS) codes, i.e., their minimum Hamming distance is \( d = n - k + 1 \). We shortly explain the interpolation problem of GS [7, 12] for list decoding GRS codes up to the Johnson radius in the following.

**Theorem 1 (Guruswami–Sudan for GRS Codes [7,12])** Let \( c \in \mathcal{GRS}(n,k) \) and \( f(X) \) the corresponding information polynomial as defined in (1). Let \( r = (r_0, \ldots, r_{n-1}) = c + e \) be a received word where \( \text{weight}(e) \leq \tau \). Let \( r'_i \) denote \( r_i/w_i \).

Let \( Q(X,Y) \in \mathbb{F}_q[X,Y] \) be a nonzero polynomial that passes through the \( n \) points \((\alpha_0, r'_0), \ldots, (\alpha_{n-1}, r'_{n-1})\) with multiplicity \( s \geq 1 \), has \( Y \)-degree at most \( \ell \), and \( \text{wdeg}_{1,k-1} Q(X,Y) < s(n - \tau) \). Then \( (Y - f(X)) \mid Q(X,Y) \).

One can easily show that a polynomial \( Q(X,Y) \) that fulfils the above conditions can be constructed whenever \( E(s, \ell, \tau) > 0 \), where

\[
E(s, \ell, \tau) = (\ell + 1)s(n - \tau) - (t+1)(k - 1) - \frac{s+1}{2}n
\]

is the difference between the maximal number of coefficients of \( Q(X,Y) \), and the number of homogeneous linear equations on \( Q(X,Y) \) specified by the interpolation constraint. This determines the maximal number of correctable errors, and one can show that satisfactory \( s \) and \( \ell \) can always be chosen whenever \( \tau < n - \sqrt{n(k-1)} \).

**Definition 2 (Permissible Triples)** An integer triple \((s, \ell, \tau) \in (\mathbb{Z}+)^3\) is permissible if \( E(s, \ell, \tau) > 0 \).

We define also the decoding radius-function \( \tau(s, \ell) \) as the greatest integer such that \((s, \ell, \tau(s, \ell))\) is permissible.

It is easy to show that \( E(s, \ell, \tau) > 0 \) for \( s > \ell \) implies \( \tau < \left\lfloor \frac{n-k}{2} \right\rfloor \), which is half the minimum distance. Therefore, it never makes sense to consider \( s > \ell \), and in the remainder we will always assume \( s \leq \ell \). Furthermore, we will also assume \( s, \ell \in O(n^2) \) since this e.g. holds for any \( \tau \) for the closed-form expressions in [7].

### 2.3 Module Reformulation of Guruswami–Sudan

Let \( M_{s,\ell} \subset \mathbb{F}_q[X,Y] \) denote the space of all bivariate polynomials passing through the points \((\alpha_0, r'_0), \ldots, (\alpha_{n-1}, r'_{n-1})\) with multiplicity \( s \) and with \( Y \)-degree at most \( \ell \). We are searching for an element of \( M_{s,\ell} \) with low \((1,k-1)\)-weighted degree.

Following the ideas of Lee and O’Sullivan [8], we can first remark that \( M_{s,\ell} \) is an \( \mathbb{F}_q[X] \) module. Second, we can give an explicit basis for \( M_{s,\ell} \). Define first two polynomials \( G(X) = \prod_{i=0}^{s-1} (X - \alpha_i) \) as well as \( R(X) \) in \( \mathbb{F}_q[X] \) as the unique Lagrange interpolation polynomial going through the points \((\alpha_i, r'_i)\) for \( i = 0, \ldots, n-1 \). Denote by \( Q[\ell](X) \) the \( Y^\ell \)-coefficient of \( Q(X,Y) \) when \( Q \) is regarded over \( \mathbb{F}_q[X][Y] \).

**Lemma 3** Let \( Q(X,Y) \in M_{s,\ell} \) with \( \deg_Y Q = t < s \). Then \( G(X)^{s-t} \mid Q[\ell](X) \).

Proof \( Q(X,Y) \) interpolates the \( n \) points \((\alpha_i, r'_i)\) with multiplicity \( s \), so for any \( i \), \( Q(X + \alpha_i, Y + r'_i) = \sum_{j=0}^{s} Q[j](X + \alpha_j)(Y + r'_j)^j \) has no monomials of total degree less than \( s \). Multiplying out the \((Y + r'_j)^j\)-terms, \( Q[j](X + \alpha_j)Y^\ell \) will be the only term with \( Y \)-degree \( t \). Therefore \( Q[j](X + \alpha_j) \) can have no monomials of degree less than \( s - t \), which implies \((X - \alpha_i) \mid Q[j](X) \). As this holds for any \( i \), we proved the lemma. \( \square \)
Theorem 4 The module $M_{s,\ell}$ is generated as an $\mathbb{F}_q[X]$-module by the $\ell+1$ polynomials $P^{(t)}(X,Y) \in \mathbb{F}_q[X,Y]$ given by
\[
P^{(t)}(X,Y) = G(X)^{s-t}(Y - R(X))^\ell, \quad \text{for } 0 \leq t < s,
P^{(t)}(X,Y) = Y^{t-s}(Y - R(X))^s, \quad \text{for } s \leq t \leq \ell.
\]

Proof It is easy to see that each $P^{(t)}(X,Y) \in M_{s,\ell}$ since both $G(X)$ and $(Y - R(X))$ go through the $n$ points $(\alpha_i, r'_i)$ with multiplicity one, and that $G(X)$ and $(Y - R(X))$ divide $P^{(t)}(X,Y)$ with total power $s$ for each $t$.

To see that any element of $M_{s,\ell}$ can be written as an $\mathbb{F}_q[X]$-combination of the $P^{(t)}(X,Y)$, let $Q(X,Y)$ be some element of $M_{s,\ell}$. Then the polynomial $Q^{(\ell-1)}(X,Y) = Q(X,Y) - Q^{(\ell)}(X,Y)$ has $Y$-degree at most $\ell - 1$. Since both $Q(X,Y)$ and $P^{(t)}(X,Y)$ are in $M_{s,\ell}$, so must $Q^{(\ell-1)}(X,Y)$ be in $M_{s,\ell}$. Since $P^{(t)}(X,Y)$ has $Y$-degree $t$ and $P^{(t)}(X) = 1$ for $t = \ell, \ell-1, \ldots, s$, we can continue reducing this way until we reach $Q^{(s-1)}(X,Y) \in M_{s,\ell}$ with $Y$-degree at most $s - 1$. From then on, we have $P^{(t)}(X) = G(X)^{s-t}$, but by Lemma 3, we must also have $G(X) | Q^{(s-1)}(X)$, so we can also reduce by $P^{(s-1)}(X,Y)$. This can be continued with the remaining $P^{(t)}(X,Y)$, eventually reducing the remainder to 0.

We can represent the basis of $M_{s,\ell}$ by the $(\ell + 1) \times (\ell + 1)$ matrix $A_{s,\ell} = \langle P^{(t)}(X,Y) \rangle_{t=0, j=0}^{t=\ell}$ over $\mathbb{F}_q[X]$. Any $\mathbb{F}_q[X]$-linear combination of rows of $A_{s,\ell}$ thus corresponds to an element in $M_{s,\ell}$ by its $t$th position being the $\mathbb{F}_q[X]$-coefficient to $Y^t$. All other bases of $M_{s,\ell}$ can be similarly represented by matrices, and these will be unimodular equivalent to $A_{s,\ell}$, i.e., they can be obtained by multiplying $A_{s,\ell}$ on the left with an invertible matrix over $\mathbb{F}_q[X]$.

Extending the work of Lee and O’Sullivan [8], Beelen and Brander [2] gave a fast algorithm for computing a satisfactory $Q(X,Y)$: start with $A_{s,\ell}$ as a basis of $M_{s,\ell}$ and compute a different, “minimal” basis of $M_{s,\ell}$ where an element of minimal $(1, k - 1)$-weighted degree appears directly.\(^1\)

In the following section, we give further details on how to compute such a basis, but our ultimate aims in Section 4 are different: we will use a minimal basis of $M_{s,\ell}$ to efficiently compute one for $M_{\hat{s},\hat{\ell}}$ for $s \geq \hat{s}$ and $\ell > \hat{\ell}$. This will allow an iterative refinement for increasing $s$ and $\ell$, where after each step we have such a minimal basis for $M_{s,\ell}$. We then exploit this added flexibility in our multi-trial algorithm.

3 Module Minimisation

Given a basis of $M_{s,\ell}$, e.g. $A_{s,\ell}$, the module minimisation here refers to the process of obtaining a new basis, which is the smallest among all bases of $M_{s,\ell}$ in a precise sense. We will define this and connect various known properties of such matrices, and use this to more precisely bound the asymptotic complexity with which they can be computed by Alekhnovich’s algorithm.

Definition 5 (Weak Popov Form [10]) A matrix $\mathcal{V}$ over $\mathbb{F}_q[X]$ is in weak Popov form if an only if the leading position of each row is different.

\(^{1}\) Actually, $A_{s,\ell}$ is the basis of [2, Remark 16], and not the main ones presented in [2,8].
We are essentially interested in short vectors in a module, and the following lemma shows that the simple concept of weak Popov form will provide this. It is a paraphrasing of [1, Proposition 2.3] and we omit the proof.

Lemma 6 (Minimal Degree) If a square matrix $V$ over $F_{q}[X]$ is in weak Popov form, then one of its rows has minimal degree of all vectors in the row space of $V$.

Denote now by $\mathcal{W}_{\ell}$ the diagonal $(\ell + 1) \times (\ell + 1)$ matrix over $F_{q}[X]$:

$$\mathcal{W}_{\ell} \triangleq \text{diag} \left( 1, X^{k-1}, \ldots, X^{\ell(k-1)} \right). \tag{3}$$

Since we seek an element of minimal $(1,k-1)$-weighted degree, we also need the following corollary.

Corollary 7 (Minimal Weighted Degree) Let $B \in F_{q}[X]^{(\ell+1) \times (\ell+1)}$ be the matrix representation of a basis of $M_{s,\ell}$. If $BW_{\ell}$ is in weak Popov form, then one of the rows of $B$ corresponds to a polynomial in $M_{s,\ell}$ with minimal $(1,k-1)$-weighted degree.

Proof Let $\tilde{B} = BW_{\ell}$. Now, $\tilde{B}$ will correspond to the basis of an $F_{q}[X]$-module $\tilde{M}$ isomorphic to $M_{s,\ell}$, where an element $Q(X,Y) \in M_{s,\ell}$ is mapped to $Q(X,X^{k-1}Y) \in \tilde{M}$. By Lemma 6, the row of minimal degree in $\tilde{B}$ will correspond to an element of $\tilde{M}$ with minimal $X$-degree. Therefore, the same row of $B$ corresponds to an element of $M_{s,\ell}$ with minimal $(1,k-1)$-weighted degree. $\square$

We introduce what will turn out to be a measure of how far a matrix is from being in weak Popov form.

Definition 8 (Orthogonality Defect [9]) Let the orthogonality defect of a square matrix $V$ over $F_{q}[X]$ be defined as $D(V) \triangleq \deg V - \deg \det V$.

Lemma 9 If a square matrix $V$ over $F_{q}[X]$ is in weak Popov form then $D(V) = 0$.

Proof Let $v_{0}, \ldots, v_{m-1}$ be the rows of $V \in F_{q}[X]^{m \times m}$ and $v_{i} = (v_{i,0}, \ldots, v_{i,m-1})$. In the alternating sum-expression for $\det V$, the term $\prod_{i=0}^{m-1} \text{LT}(v_{i})$ will occur since the leading positions of $v_{i}$ are all different. Thus $\deg \det V = \sum_{i=0}^{m-1} \deg \text{LT}(v_{i}) = \deg V$ unless leading term cancellation occurs in the determinant expression. However, no other term in the determinant has this degree: regard some (unsigned) term in $\det V$, say $t = \prod_{i=0}^{m-1} v_{i,\sigma(i)}$ for some permutation $\sigma \in S_{m}$. If not $\sigma(i) = \text{LP}(v_{i})$ for all $i$, then there must be an $i$ such that $\sigma(i) > \text{LP}(v_{i})$ since $\sum_{i} \sigma(j)$ is the same for all $\sigma \in S_{m}$. Thus, $\deg v_{i,\sigma(i)} < \deg v_{i,\text{LP}(v_{i})}$. As none of the other terms in $t$ can have greater degree than their corresponding row’s leading term, we get $\deg t < \sum_{i=0}^{m-1} \deg \text{LT}(v_{i})$. Thus, $D(V) = 0$. $\square$

Alekhnovich [1] gave a fast algorithm for transforming a matrix over $F_{q}[X]$ to weak Popov form. For the special case of square matrices, a finer description of its asymptotic complexity can be reached in terms of the orthogonality defect, and this is essential for our decoder.
Lemma 10 (Alekhnovich’s Row-Reducing Algorithm [1]) Alekhnovich’s algorithm inputs a matrix \( V \in \mathbb{F}_q[X]^{m \times m} \) and outputs a unimodular equivalent matrix which is in weak Popov form. Let \( N \) be the greatest degree of a term in \( V \). If \( N \in O(D(V)) \) then the algorithm has asymptotic complexity:

\[
O\left( m^3 D(V) \log^2 D(V) \log \log D(V) \right) \text{ operations over } \mathbb{F}_q.
\]

Proof The description of the algorithm as well as proof of its correctness can be found in [1]. We only prove the claim on the complexity. The method \( R(V, t) \) of [1] computes a unimodular matrix \( U \) such that \( \deg(UV) \leq \deg V - t \) or \( UV \) is in weak Popov form. According to [1, Lemma 2.10], the asymptotic complexity of this computation is in \( O(m^3 t \log^2 t \log \log t) \). Due to Lemma 9, we can set \( t = D(V) \) to be sure that \( UV \) is in weak Popov form. What remains is just to compute the product \( UV \). Due to [1, Lemma 2.8], each entry in \( U \) can be represented as \( p(X)X^d \) for some \( d \in \mathbb{N}_0 \) and \( p(X) \in \mathbb{F}_q[X] \) of degree at most \( 2t \). If therefore \( N \in O(D(V)) \), the complexity of performing the matrix multiplication using the naive algorithm is \( O(m^3 D(V)) \). \( \square \)

4 Multi-Trial List Decoding

4.1 Basic Idea

Using the results of the preceding section, we show in Section 4.2 that given a basis of \( M_{s, \ell} \) as a matrix \( B_{s, \ell} \) in weak Popov form, then we can write down a matrix \( C_{s, \ell+1} \) which is a basis of \( M_{s, \ell+1} \) and whose orthogonality defect is much lower than that of \( A_{s, \ell+1} \). This means that reducing \( C_{s, \ell+1} \) to weak Popov form using Alekhnovich’s algorithm is faster than reducing \( A_{s, \ell+1} \). We call this kind of refinement a “micro-step of type I”. In Section 4.3, we similarly give a way to refine a basis of \( M_{s, \ell} \) to one of \( M_{s+1, \ell+1} \), and we call this a micro-step of type II.

If we first compute a basis in weak Popov form of \( M_{1,1} \) using \( A_{1,1} \), we can perform a sequence of micro-steps of type I and II to compute a basis in weak Popov form of \( M_{s, \ell} \) for any \( s, \ell \) with \( \ell \geq s \). After any step, having some intermediate \( \hat{s} \leq s, \hat{\ell} \leq \ell \), we will thus have a basis of \( M_{\hat{s}, \hat{\ell}} \) in weak Popov form. By Corollary 7, we could extract from \( B_{s, \ell} \) a \( Q(X, Y) \in M_{s, \ell} \) with minimal \((1, k-1)\)-weighted degree. Since it must satisfy the interpolation conditions of Theorem 1, and since the weighted degree is minimal among such polynomials, it must also satisfy the degree constraints for \( \bar{\tau} = \tau(\hat{s}, \hat{\ell}) \). By that theorem any codeword with distance at most \( \bar{\tau} \) from \( r \) would then be represented by a root of \( Q(X, Y) \).

Algorithm 1 is a generalisation and formalisation of this method. For a given \( \mathcal{G}_{\mathcal{R}_{S}}(n, k) \) code, one chooses ultimate parameters \( (s, \ell, \tau) \) being a permissible triple with \( s \leq \ell \). One also chooses a list of micro-steps and chooses after which micro-steps to attempt decoding; these choices are represented by a list of \( S_1, S_2 \) and \( \text{Root} \) elements. This list must contain exactly \( s - \ell \) \( S_1 \)-elements and \( s - 1 \) \( S_2 \)-elements, as it begins by computing a basis for \( M_{1,1} \) and will end with a basis for \( M_{s, \ell} \). Whenever there is a \( \text{Root} \) element in the list, the algorithm finds all codewords with distance at most \( \bar{\tau} = \tau(\hat{s}, \hat{\ell}) \) from \( r \); if this list is non-empty, the computation breaks and the list is returned.

The algorithm calls sub-functions which we explain informally: \( \text{MicroStep1} \) and \( \text{MicroStep2} \) will take \( s, \ell \) and a basis in weak Popov form for \( M_{s, \ell} \) and return a basis
in weak Popov form for $M_{\hat{s},\hat{\ell}+1}$ respectively $M_{\hat{s}+1,\hat{\ell}+1}$; more detailed descriptions for these are given in Subsections 4.2 and 4.3. \texttt{MinimalWeightedRow} finds a polynomial of minimal $(1,k-1)$-weighted degree in $M_{\hat{s},\hat{\ell}+1}$ given a basis in weak Popov form (Corollary 7). Finally, \texttt{RootFinding}($Q,\tau$) returns all $Y$-roots of $Q(X,Y)$ of degree less than $k$ and whose corresponding codeword has distance at most $\tau$ from the received word $r$.

\texttt{Algorithm 1: Multi-Trial Guruswami–Sudan Decoding}

\textbf{Input:}
A GRS($n,k$) code with $w_0,\ldots,w_{n-1}$
The received vector $r = (r_0,\ldots,r_{n-1})$
A permissible triple ($s,\ell,\tau$)
A list $C$ with elements in \{$S_1,S_2,\text{Root}$\} with $\ell-s$ instances of $S_1$ and $s-1$ instances of $S_2$

\textbf{Preprocessing:}
Calculate $r'_i = r_i/w_i$ for all $i = 0,\ldots,n-1$
Construct $A_{1,1}$, and compute $B_{1,1}$ from $A_{1,1}W_1$ using Alekhnovich’s algorithm
Initial parameters ($\hat{s},\hat{\ell}$) $\leftarrow$ (1,1)

\hspace{0.5cm}1 for each $c$ in $C$ do
\hspace{1.5cm}2 \hspace{0.5cm}if $c = S_1$ then
\hspace{2.5cm}3 \hspace{0.5cm}if $c = S_1$ then
\hspace{3.5cm}4 \hspace{0.5cm}$B_{\hat{s},\hat{\ell}+1} \leftarrow \text{MicroStep1}(\hat{s},\hat{\ell},B_{\hat{s},\hat{\ell}})$
\hspace{3.5cm}5 \hspace{0.5cm}$(\hat{s},\hat{\ell}) \leftarrow (\hat{s},\hat{\ell}+1)$
\hspace{1.5cm}6 \hspace{0.5cm}if $c = S_2$ then
\hspace{2.5cm}7 \hspace{0.5cm}if $c = S_2$ then
\hspace{3.5cm}8 \hspace{0.5cm}$B_{\hat{s}+1,\hat{\ell}+1} \leftarrow \text{MicroStep2}(\hat{s},\hat{\ell},B_{\hat{s},\hat{\ell}})$
\hspace{3.5cm}9 \hspace{0.5cm}$(\hat{s},\hat{\ell}) \leftarrow (\hat{s}+1,\hat{\ell}+1)$
\hspace{1.5cm}10 \hspace{0.5cm}if $c = \text{Root}$ then
\hspace{2.5cm}11 \hspace{0.5cm}if $c = \text{Root}$ then
\hspace{3.5cm}12 \hspace{0.5cm}$Q(X,Y) \leftarrow \text{MinimalWeightedRow}(B_{\hat{s},\hat{\ell}})$
\hspace{3.5cm}13 \hspace{0.5cm}if \text{RootFinding}($Q(X,Y),\tau(\hat{s},\hat{\ell})$) $\neq \emptyset$ then
\hspace{3.5cm}14 \hspace{0.5cm}return this list

The correctness of Algorithm 1 for any possible choice of $s,\ell$ and $C$ follows from our discussion as well as Sections 4.2 and 4.3.

Since we can only perform micro-steps of type I and II, there is inherent limitations in the “reachable” permissible triples. Firstly, we can never choose $s > \ell$; but as mentioned in Section 2, such a choice never makes sense. On the other hand, if we first wish to attempt decoding at some permissible ($s_1,\ell_1,\tau_1$) and thereafter continue to ($s_2,\ell_2,\tau_2$) with $\tau_2 > \tau_1$ in case of decoding failure, then it might be the case that either $s_1 > s_2, \ell_1 > \ell_2$ or $s_2 - s_1 > \ell_2 - \ell_1$, each of these precluding the possibility of reaching the latter triple from the former by micro-steps of type I and II. We conjecture, however, based on experimentation, that if each triple is selected such that the multiplicity is minimal for the given decoding radius, and the list size is chosen minimal under these two constraints, then this will never be the case; in other words, successions of “sensibly” chosen triples will always be reachable by a sequence of micro-steps of type I and II.

In the following two subsections we explain the details of the micro-steps. In Section 4.4, we discuss the complexity of the method and how the choice of $C$ influence this.
4.2 Micro-Step Type I: \((s, \ell) \mapsto (s, \ell + 1)\)

**Lemma 11** If \(B^{(0)}(X, Y), \ldots, B^{(\ell)}(X, Y) \in \mathbb{F}_q[X, Y]\) is a basis of \(M_{s, \ell}\), then the following is a basis of \(M_{s, \ell + 1}\):

\[
B^{(0)}(X, Y), \ldots, B^{(\ell)}(X, Y), Y^{\ell-s+1}(Y - R(X))^s.
\]

**Proof** In the basis of \(M_{s, \ell + 1}\) given in Theorem 4, the first \(\ell + 1\) generators are the generators of \(M_{s, \ell}\). Thus all of these can be described by any basis of \(M_{s, \ell + 1}\). The last remaining generator is exactly \(Y^{\ell-s+1}(Y - R(X))^s\). \(\Box\)

In particular, the above lemma holds for a basis of \(M_{s, \ell + 1}\) in weak Popov form, represented by a matrix \(B_{s, \ell}\). The following matrix thus represents a basis of \(M_{s, \ell + 1}\):

\[
C_{s, \ell + 1}^1 = \begin{bmatrix}
B_{s, \ell} & 0^T \\
0 \ldots 0 & (-R)^s \begin{pmatrix} 1 & \ell \end{pmatrix}(-R)^{s-1} \ldots 1
\end{bmatrix}.
\]

(4)

**Lemma 12** \(D(c^1_{s, \ell + 1}W_{\ell + 1}) = s(\deg R - k + 1) \leq s(n - k)\).

**Proof** We calculate the two quantities \(\det(c^1_{s, \ell + 1}W_{\ell + 1})\) and \(\deg(c^1_{s, \ell + 1}W_{\ell + 1})\). It is easy to see that

\[
\det(c^1_{s, \ell + 1}W_{\ell + 1}) = \det B_{s, \ell} \det W_{\ell + 1} = \det B_{s, \ell} \det W_{X}(\ell+1)(k-1).
\]

For the row-degree, it is clearly \(\deg(B_{s, \ell}W_{\ell})\) plus the row-degree of the last row. If and only if the received word is not a codeword then \(\deg R \geq k\), so the leading term of the last row must be \((-R)^sX^{(\ell+1-s)}(k-1)\). Thus, we get

\[
D(c^1_{s, \ell + 1}W_{\ell + 1}) = (\deg(B_{s, \ell}W_{\ell}) + s \deg R + (\ell + 1 - s)(k - 1))
- (\deg \det(B_{s, \ell}W_{\ell}) + (\ell + 1)(k - 1))
= s(\deg R - k + 1),
\]

where the last step follows from Lemma 9 as \(B_{s, \ell}W_{\ell}\) is in weak Popov form. \(\Box\)

**Corollary 13** The complexity of MicroStep1\((s, \ell, B_{s, \ell})\) is \(O(\ell^3 sn \log^2 n \log \log n)\).

**Proof** Follows by Lemma 10. Since \(s \in O(n^2)\) we can leave out the \(s\) in log-terms. \(\Box\)

4.3 Micro-Step Type II: \((s, \ell) \mapsto (s + 1, \ell + 1)\)

**Lemma 14** If \(B^{(0)}(X, Y), \ldots, B^{(\ell)}(X, Y) \in \mathbb{F}_q[X, Y]\) is a basis of \(M_{s, \ell}\), then the following is a basis of \(M_{s + 1, \ell + 1}\):

\[
G^{s+1}(X), B^{(0)}(X, Y)(Y - R(X)), \ldots, B^{(\ell)}(X, Y)(Y - R(X)).
\]
Proof Denote by $P^{(0)}(X, Y), \ldots, P^{(t)}_{s, \ell}(X, Y)$ the basis of $M_{s, \ell}$ as given in Theorem 4, and by $P^{(0)}_{s+1, \ell+1}(X, Y), \ldots, P^{(t)}_{s+1, \ell+1}(X, Y)$ the basis of $M_{s+1, \ell+1}$. Then observe that for $t > 0$, we have $P^{(t)}_{s+1, \ell+1} = P^{(t)}_{s, \ell}(Y - R(X))$. Since the $B^{(i)}(X, Y)$ form a basis of $M_{s, \ell}$, each $P^{(t)}_{s, \ell}$ is expressible as an $F_q[X]$-combination of these, and thus for $t > 0$, $P^{(t)}_{s+1, \ell+1} = P^{(t)}_{s, \ell}(Y - R(X))$. Remaining is then only $P^{(0)}_{s+1, \ell+1}(X, Y) = G^{s+1}(X)$. □

As before, we can use the above with the basis $B_{s, \ell}$ of $M_{s, \ell}$ in weak Popov form, found in the previous iteration of our algorithm. Remembering that multiplying by $Y$ translates to shifting one column to the right in the matrix representation, the following matrix thus represents a basis of $M_{s+1, \ell+1}$:

$$C^{II}_{s+1, \ell+1} = \begin{bmatrix}
G^{s+1} & 0 \\
0^T & 0
\end{bmatrix} + \begin{bmatrix}
0 \\
0^T & B_{s, \ell}
\end{bmatrix} - R \cdot \begin{bmatrix}
0 \\
B_{s, \ell} & 0^T
\end{bmatrix}. \quad (5)$$

Lemma 15 $D(C^{II}_{s+1, \ell+1} W_{\ell+1}) = (\ell + 1)(\deg R - k + 1) \leq (\ell + 1)(n - k)$.

Proof We compute $\deg D(C^{II}_{s+1, \ell+1} W_{\ell+1})$ and $\deg \det(C^{II}_{s+1, \ell+1} W_{\ell+1})$. For the former, obviously the first row has degree $(s + 1)n$. Let $b_i$ denote the $i$th row of $B_{s, \ell}$ and $b'_i$ denote the $i$th row of $B_{s, \ell} W_{\ell}$. The $(i + 1)$th row of $C^{II}_{s+1, \ell+1} W_{\ell+1}$ has the form

$$[(0 | b_i) - R(b_i | 0)] W_{\ell+1} = (0 | b'_i) X^{k-1} - R(b'_i | 0).$$

If and only if the received word is not a codeword, then $\deg R \geq k$. In this case, the leading term of $Rb'_i$ must have greater degree than any term in $X^{k-1}b'_i$. Thus the degree of the above row is $\deg R + \deg b'_i$. Summing up we get

$$\deg C^{II}_{s+1, \ell+1} = (s + 1)n + \sum_{i=0}^\ell \deg R + \deg b'_i$$

$$= (s + 1)n + (\ell + 1) \deg R + \deg(B_{s, \ell} W_{\ell}).$$

For the determinant, observe that

$$\det(C^{II}_{s+1, \ell+1} W_{\ell+1}) = \det(C^{II}_{s+1, \ell+1}) \det(W_{\ell+1})$$

$$= G^{s+1} \det \tilde{B} \det W_{\ell} X^{(\ell+1)(k-1)},$$

where $\tilde{B} = B_{s, \ell} - R \begin{bmatrix} B_{s, \ell} & 0^T \end{bmatrix}$ and $\tilde{B}_{s, \ell}$ is all but the zeroth column of $B_{s, \ell}$. This means $\tilde{B}$ can be obtained by starting from $B_{s, \ell}$ and iteratively adding the $(j + 1)$th column of $B_{s, \ell}$ scaled by $R(X)$ to the $j$th column, with $j$ starting from $0$ up to $\ell - 1$. Since each of these will add a scaled version of an existing column in the matrix, this does not change the determinant. Thus, $\det \tilde{B} = \det B_{s, \ell}$. But then $\det \tilde{B} \det W_{\ell} = \det(B_{s, \ell} W_{\ell})$ and so $\deg(\det \tilde{B} \det W_{\ell}) = \deg(B_{s, \ell} W_{\ell})$ by Lemma 9 since $B_{s, \ell} W_{\ell}$ is in weak Popov form. Thus we get

$$\deg \det(C^{II}_{s+1, \ell+1} W_{\ell+1}) = (s + 1)n + \deg(B_{s, \ell} W_{\ell}) + (\ell + 1)(k - 1).$$

The lemma follows from the difference of the two calculated quantities. □

Corollary 16 The complexity of MicroStep2$(s, \ell, B_{s, \ell})$ is $O(\ell^4 n \log^2 n \log \log n)$. 
4.4 Complexity Analysis

Using the estimates of the two preceding subsections, we can make a rather precise worst-case asymptotic complexity analysis of Algorithm 1. The average running time will depend on the exact choice of $C$ but we will see that the worst-case complexity will not. First, it is necessary to know the complexity of performing a root-finding attempt.

**Lemma 17 (Complexity of Root-Finding)** Given a polynomial $Q(X,Y) \in \mathbb{F}_q[X][Y]$ of $Y$-degree at most $\ell$ and $X$-degree at most $N$, there exists an algorithm to find all $\mathbb{F}_q[X]$-roots of complexity $O(\ell^2N \log^2 N \log \log N)$, assuming $\ell, q \in O(N)$.


**Theorem 18 (Complexity of Algorithm 1)** For a given GRS$(n,k)$ code, as well as a given list of steps $C$ for Algorithm 1 with ultimate parameters $(s, \ell, \tau)$, the algorithm has worst-case complexity $O(\ell^4sn \log^2 n \log \log n)$, assuming $q \in O(n)$.

**Proof** The worst-case complexity corresponds to the case that we do not break early but run through the entire list $C$. Precomputing $A_{\hat{s}, \hat{\ell}}$ using Lagrangian interpolation can be performed in $O(n \log^2 n \log \log n)$, see e.g. [5, p. 235], and reducing to $B_{s, \ell}$ is in the same complexity by Lemma 10.

Now, $C$ must contain exactly $\ell - s$ $S_1$-elements and $s - 1$ $S_2$-elements. The complexities given in Corollaries 13 and 16 for some intermediate $\hat{s}, \hat{\ell}$ can be relaxed to $s$ and $\ell$. Performing $O(\ell)$ micro-steps of type I and $O(s)$ of type II is therefore in $O(\ell^4sn \log^2 n \log \log n)$.

It only remains to count the root-finding steps. Obviously, it never makes sense to have two $\text{Root}$ after each other in $C$, so after removing such possible duplicates, there can be at most $\ell$ elements $\text{Root}$. When we perform root-finding for intermediate $\hat{s}, \hat{\ell}$, we do so on a polynomial in $M_{\hat{s}, \hat{\ell}}$ of minimal weighted degree, and by the definition of $M_{\hat{s}, \hat{\ell}}$ as well as Theorem 1, this weighted degree will be less than $\hat{s}(n - \hat{\tau}) < sn$. Thus we can apply Lemma 17 with $N = sn$. $\square$

The worst-case complexity of our algorithm is equal to the average-case complexity of the Beelen–Brander [2] list decoder. However, Theorem 18 shows that we can choose as many intermediate decoding attempts as we would like without changing the worst-case complexity. One could therefore choose to perform a decoding attempt just after computing $B_{1,1}$ as well as every time the decoding radius has increased. The result would be a decoding algorithm finding all closest codewords within some ultimate radius $\tau$. If one is working in a decoding model where such a list suffices, our algorithm will thus have much better average-case complexity since fewer errors occur much more frequently than many.

5 Conclusion

An iterative interpolation procedure for list decoding GRS codes based on Alekhnovich’s module minimisation was proposed and shown to have the same worst-case complexity as Beelen and Brander’s [2]. We showed how the target module used
in Beelen–Brander can be minimised in a progressive manner, starting with a small module and systematically enlarging it, performing module minimisation in each step. The procedure takes advantage of a new, slightly more fine-grained complexity analysis of Alekhnovich’s algorithm, which implies that each of the module refinement steps will run fast.

The main advantage of the algorithm is its granularity which makes it possible to perform fast multi-trial decoding: we attempt decoding for progressively larger decoding radii, and therefore find the list of codewords closest to the received. This is done without a penalty in the worst case but with an obvious benefit in the average case.

The decoding algorithm presented here has been implemented and verified in Sage version 5.3. The source code is available upon demand.

Acknowledgement

The authors thank Daniel Augot for fruitful discussions. This work has been supported by German Research Council Deutsche Forschungsgemeinschaft (DFG) under grant BO 867/22-1. Johan S. R. Nielsen also gratefully acknowledges the support from The Otto Monsted Foundation and the Idella Foundation, as well as the support from the Danish National Research Foundation and the National Science Foundation of China (Grant No.11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography.

References

Some bounds on the size of codes

Emanuele Bellini · Eleonora Guerrini · Massimiliano Sala

Received: date / Accepted: date

Abstract We present some upper bounds on the size of non-linear codes and their restriction to systematic codes and linear codes. These bounds are independent of other known theoretical bounds, e.g. the Griesmer bound, the Johnson bound or the Plotkin bound, and one of these is actually an improvement of a bound by Zinoviev, Litsyn and Laihonen. Our experiments show that in some cases (the majority of cases for some $q$) our bounds provide the best value, compared to all other theoretical bounds.

Keywords Hamming distance · linear code · systematic code · non-linear code · upper bound

1 Introduction

The problem of bounding the size of a code depends heavily on the code family that we are considering. In this paper we are interested in three types of codes: linear codes, systematic codes and non-linear codes. Referring to the subsequent section for rigorous definitions, with linear codes we mean linear subspaces of $(F_q)^n$, while with non-linear codes we mean (following consolidated tradition) codes that are not necessarily linear. Systematic codes form a less-studied family of codes, whose definition is given in the next section. The size of a systematic code is directly comparable with that of a linear code, since it is a power of the field size.
In this paper we are interested only in theoretical bounds, that is, bounds on the size of a code that can be obtained by a closed-formula expression, although other algorithmic bounds exist (e.g. the Linear Programming bound [Del73]).

The academic literature reports only one bound for linear codes, the Griesmer bound ([Gri60]), no bounds for systematic codes and many bounds for non-linear codes. Among those, we recall: the Johnson bound ([Joh62],[Joh71],[HP03]), the Elias-Bassalygo bound ([Bas65],[HP03]), the Levenshtein bound ([Lev98]), the Hamming (Sphere Packing) bound and the Singleton bound ([PBH98]), and the Plotkin bound ([Plo60], [HP03]).

Since the Griesmer bound is specialized for linear codes, we would expect it to beat the other bounds, but even this does not happen, except in some cases. So we have an unexpected situation where the bounds holding for the more general case are numerous and beat bounds holding for the specialized case.

In this paper we present one (closed-formula) bound (Bound A) for a large part of non-linear codes (including all systematic codes), which is an improvement of a bound by Zinoviev, Litsyn and Laihonen ([ZL84], russian, and [LL98]). The crux of our improvement is a preliminary result presented in Section 2, while in Section 3 we are able to prove Bound A. Then we restrict Bound A to the systematic/linear case and compare it with all the before-mentioned bounds by computing their values for a large set of parameters (corresponding to about one week of computations with our computers). Our findings are in favour of Bound A and are reported in Section 4. For large values of q, our bound provides the best value in the majority of cases.

The only bound that we never beat is Plotkin’s, but its range is very small (the distance has to be at least \( d > n(1 - 1/q) \)) and the cases falling in this range are a tiny portion with large q’s.

For standard definitions and known bounds, the reader is directed to the original articles or to any recent good book, e.g. [HP03] or [PBH98].

2 Preliminaries

We first recall a few definitions.

Let \( \mathbb{F}_q \) be the finite field with q elements, where q is any power of any prime.

Let \( n \geq k \geq 1 \) be integers. Let \( C \subseteq \mathbb{F}_q^n \), \( C \neq \emptyset \). We say that \( C \) is an \((n, q)\) \textit{code}. Any \( c \in C \) is a \textit{word}. Note that here and afterwards a “code” denotes what is called a “non-linear code” in the introduction.

Let \( \phi : (\mathbb{F}_q)^k \to (\mathbb{F}_q)^n \) be an injective function and let \( C = \text{Im}(\phi) \). We say that \( C \) is an \((n, k, q)\) \textit{systematic code} if \( \phi(v)_i = v_i \) for any \( v \in (\mathbb{F}_q)^k \) and any \( 1 \leq i \leq k \).

If \( C \) is a vector subspace of \((\mathbb{F}_q)^n\), then \( C \) is a \textit{linear} code. Clearly any non-zero linear code is equivalent to a systematic code.

From now on, \( \mathbb{F} \) will denote \( \mathbb{F}_q \) and \( q \) is understood.

We denote with \( d(c, c') \) the \textbf{(Hamming) distance} of two words \( c, c' \in C \), which is the number of different components between \( c \) and \( c' \). We denote with \( d \) a number such that \( 1 \leq d \leq n \) to indicate the \textbf{(Hamming) distance of a code}, which is \( d = \min_{c, c' \in C, c \neq c'} \{d(c, c')\} \). Note that a code with only one word has, by convention, distance equal to infinity. The whole \( \mathbb{F}_q^n \) has distance 1, and \( d = n \) in a systematic code is possible only if \( k = 1 \).

From now on, \( n, k \) are understood.
Consider the distance from \( \rho \), we start with proving \( |C| \leq |D| \) and \( |B_0(d + \epsilon - 1, n) \). Let \( \rho = d - 1 \) and \( r = d + \epsilon - 1 \), so that \( r - \rho = \epsilon \), and let \( N = D \cap B_0(r, n) \). We have: \( D = D \setminus N, |D| = |\bar{D}| - |N| \).

We are searching for a lower bound on \( |N| \), in order to have an upper bound on \( |D| \). We start with proving \( B_0(r - \rho, n) \subseteq \bigcup_{x \in N} B_x(\rho, n) \).

Consider \( y \in B_0(r - \rho, n) \). If for all \( x \in N \) we have that \( y \notin B_x(\rho, n) \), then \( y \) is a vector whose distance from \( N \) is at least \( \rho + 1 \). Since \( y \in B_0(r - \rho, n) \), also its distance from \( D \setminus N \) is at least \( \rho + 1 \). Therefore, the distance of \( y \) from the whole \( D \) is at least \( \rho + 1 = d \) and so we can obtain a new code \( \bar{D} \cup \{ y \} \) containing \( D \) and with distance \( d \), contradicting the fact that \( |\bar{D}| \) is the largest size for such a code in \( F^n \).

So, our claim must hold, and its consequence is \( |N| \cdot |B_x(\rho, n)| \geq |B_0(r - \rho, n)| \), which gives \( |N| \geq \frac{|B_0(r - \rho, n)|}{|B_x(\rho, n)|} = \frac{|B_0(r - \rho, n)|}{|B_x(d - 1, n)|} \).

Using previous observations, we obtain the desired bound:

\[ |C| \leq |\bar{D}| - |\bar{D} \cap B_0(d + \epsilon - 1, n)| \leq A_q(n, d) - \frac{|B_x(\rho, n)|}{|B_x(d - 1, n)|}. \]

3 An improvement of the Zinoviev-Litsyn-Laihonen bound

In 1998 Litsyn and Laihonen prove a bound for non-linear codes: Theorem 1 of [LL98], which we write with our notation as follows.

**Theorem 1 (Zinoviev-Litsyn-Laihonen bound)** Let \( 1 \leq d \leq n \). Let \( t \in \mathbb{N} \) be such that \( t \leq n - d \). Let \( r \in \mathbb{N} \) be such that \( d - 2r \leq n - t \), \( 0 \leq r \leq t \) and \( 0 \leq r \leq \frac{1}{2}d \). Then

\[ A_q(n, d) \leq \frac{q^t}{|B(r, t)|} A_q(n - t, d - 2r). \]
Let $C$ be an $(n, d)$-code over $F$, let $k = \lfloor \log_q(|C|) \rfloor$. We say that $C$ is systematic-embedding if $C$ contains a systematic code $D$ with size $|D| = q^k$. Obviously a systematic code is systematic-embedding with $D = C$. Moreover if the code is linear then $k$ is the dimension of $C$.

All known families of maximal codes are either systematic codes or systematic-embedding codes (see e.g., [Pre68], [Ker72] and [BvLW83]).

We are ready to show a strengthening of Theorem 1 restricted to systematic-embedding codes: Bound $A$. In the proof we follow initially the outline of the proof of [LL98][Theorem 1] and then we apply Proposition 1.

**Theorem 2 (Bound $A$)** Let $1 \leq d \leq n$. Let $t \in \mathbb{N}$ be such that $t \leq n - d$. Let $r \in \mathbb{N}$ be such that $d - 2r \leq n - t$, $0 \leq r \leq t$ and $0 \leq r \leq \frac{1}{2}d$. Suppose that there is an $(n, d)$-code $C$ over $F$ such that $|C| = A_d(n, d)$ and $C$ is systematic-embedding. Let $t \leq k = \lfloor \log_q(|C|) \rfloor$. Then

\[
A_q(n, d) \leq \frac{q^t}{|B(r, t)|} \left( A_q(n - t, d - 2r) - \frac{|B(r, n - t)|}{|B(d - 2r - 1, n - t)|} + 1 \right).
\]

**Proof** We consider an $(n, d)$ code $C$ such that $|C| = A_q(n, d)$ and $C$ is systematic-embedding. By hypothesis $C$ must exist. We number all words in $C$ in any order: $C = \{c_i \mid 1 \leq i \leq A_q(n, d)\}$. We indicate the $i$-th word with $c_i = (c_{1,i}, \ldots, c_{k,i})$. We puncture $C$ as follows:

(i) we choose any $t$ columns among the $k$ columns of the systematic part of $C$, $1 \leq j_1, \ldots, j_t \leq n$; since two codes are equivalent w.r.t. column permutations we suppose $j_1 = 1, \ldots, j_t = t$.

Let us split each word $c_i \in C$ in two parts:

\[c_i = (c_{1,i}, \ldots, c_{j_i}), \tilde{c}_i = (c_{j_i+1}, \ldots, c_{k,i})\]

so $c_i = (\tilde{c}_i, c_{j_i})$.

(ii) We choose a $z \in F^t$.

(iii) We collect in $I$ all $i$'s s.t. $d(z, \tilde{c}_i) \leq r$.

(iv) We delete the first $t$ components of $\{c_i \mid i \in I\}$.

Then the punctured code $\tilde{C}_2$ obtained by (i),(ii),(iii) and (iv) is:

\[\tilde{C}_2 = \{\tilde{c}_i \mid i \in I\} = \{\tilde{c}_i \mid 1 \leq i \leq A_q(n, d), d(z, \tilde{c}_i) \leq r\}.
\]

We claim that we can choose $z$ in such a way that $\tilde{C}_2$ is equivalent to a code with the following properties:

\[
\begin{align*}
\bar{n} &= n - t \quad \text{(1)} \\
\bar{d} &\geq d - 2r \quad \text{(2)} \\
|\tilde{C}_2| &\geq \frac{|C|}{q^t} |B(r, t)| \quad \text{(3)} \\
\text{w}(\tilde{c}_i) &\geq d - r \quad \text{for all } \tilde{c}_i \neq 0. \quad \text{(4)}
\end{align*}
\]

(1) is obvious. As regards (2), note that $d(c_i, c_j) = d(\tilde{c}_i, \tilde{c}_j) + d(c_{j_i}, c_{j_j}) \geq d$ and also that $\tilde{c}_i, \tilde{c}_j \in B_2(r, t)$ implies $d(\tilde{c}_i, \tilde{c}_j) \leq 2r$. Therefore for any $i \neq j$ $2r + d(c_{j_i}, c_{j_j}) \geq d(\tilde{c}_i, \tilde{c}_j) + d(c_{j_i}, c_{j_j}) \geq d$.

The proof of (3) is more involved and we need to consider the average number $M$ of the $i$'s such that $\tilde{c}_i$ happens to be in a sphere of radius $r$ (in $F^t$). The average is taken over all sphere centers, that is, all vectors $x$'s in $F^t$, so that $M = \frac{1}{|F^t|} \sum_{x \in F^t} |\{i \mid 1 \leq i \leq A_q(n, d), \tilde{c}_i \in B_2(r, t)\}|.$
Remark 1

Theorem 2 and Corollary 1 can be clearly generalized to any alphabet. Our bound then reduces to: $0 \leq Litsyn-Laihonen$ bound, which, when $|\mathcal{A}| = 0$.

If $\bar{x}$ is a word of weight at least $c$ such that $\bar{x} \in \mathcal{C}$, we can write $M = \frac{1}{q^n} \sum_{\bar{x} \in \mathcal{F}^n} \psi(x, \bar{x})$. By swapping variables we get:

$$M = \frac{1}{q^n} \sum_{\bar{x} \in \mathcal{F}^n} \sum_{i=1}^{A_q(n,d)} \psi(x, \bar{x}_i) = \frac{1}{q^n} \sum_{i=1}^{A_q(n,d)} \sum_{\bar{x} \in \mathcal{F}^n} \psi(x, \bar{x}_i) = A_q(n,d) \frac{M}{q^n} |B(r, t)|.$$

This means that there exists $\bar{x} \in \mathcal{F}^n$ such that:

$$\{i \mid 1 \leq i \leq A_q(n,d), \bar{x}_i \in B_2(r, t)\} \geq M \geq \frac{A_q(n,d)}{q^n} |B(r, t)|.$$

In other words, there are at least $\frac{A_q(n,d)}{q^n} |B(r, t)|$ of $\bar{x}_i$'s such that their corresponding $\bar{c}_i$'s are contained in $B_2(r, t)$. Distinct $\bar{c}_i$'s may well give rise to the same $\bar{c}_i$'s, but they always correspond to distinct $\bar{c}_i$'s (see the proof of (2)), so there are at least $\frac{|C|}{q^n} |B(r, t)|$ (distinct) $\bar{c}_i$'s such that their corresponding $\bar{c}_i$'s fall in $B_2(r, t)$. By choosing $z = \bar{x}$ we then have at least $\frac{|C|}{q^n} |B(r, t)|$ (distinct) codewords of $\mathcal{C}$, and so (3) follows.

We claim that (4) holds if $0 \in \mathcal{C}$ and $z = 0$. Indeed: $w(c) = d(0, c) \geq d, \forall c \in \mathcal{C}$ such that $c \neq 0$, and $z = 0 \iff y \in B_2(r, t) \iff w(y) \leq r$.

As a consequence, any nonzero word $c_i = (\bar{c}_i, \bar{c}_i)$ of weight at most $r$ in $\mathcal{C}$ has weight at least $d - r$ in the other $n - t$ components. If $0 \notin \mathcal{C}$ or $z \neq 0$ we consider a code $C + v$ equivalent to $C$, by choosing the translation $v$ in the following way. By hypothesis of systematic-embedding there exists $\bar{c} \in \mathcal{C}$ such that its first $t$ coordinates form the vector $\bar{x}$. By choosing $v = \bar{c}$ we obtain the desired code, thus (4) is proved.

Now we call $X$ the largest $(\bar{n}, d - 2r)$-code containing the zero word and such that $w(\bar{x}) \geq d - r = (d - 2r) + r, \forall \bar{x} \in X$. Observe that $X$ satisfies (1), (2), (3), (4) and so $|X| \geq |\mathcal{C}|$. Then we can apply Proposition 1 to $X \setminus \{0\}$ and $\epsilon = r$, and obtain the following chain of inequalites:

$$\frac{|C|}{q^n} |B(r, t)| \leq |\mathcal{C}| \leq |X| \leq A_q(\bar{n}, d - 2r) - \frac{|B(r, n)|}{|B(d - 2r - 1, n)|} + 1, \text{ and since } |\mathcal{C}| = A_q(n, d)$$

we have the bound: $A_q(n, d) \leq \frac{q^n}{|B(r, n)|} \left( A_q(\bar{n}, d - 2r) - \frac{|B(r, n)|}{|B(d - 2r - 1, n)|} + 1 \right)$.

3.1 Systematic case

When we restrict ourselves into the systematic/linear case, then the value $A_q(n, d)$ can only be a power of $q$, and if the dimension of the code $C$ is $k$, then $A_q(n, d) = q^k$.

By choosing $t = k$ we have the following corollary:

**Corollary 1 (Bound B)** Let $k, d, r \in \mathbb{N}, d \geq 2, k \geq 1$. Let $n$ be such that there exists an $(n, k, q)$ systematic code $C$ with distance at least $d$.

If $0 \leq r < \min\{d, \left[\frac{q^k}{2}\right]\}$, then:

$$|B(r, k)| \leq A_q(n - k, d - 2r) - \frac{|B(r, n-k)|}{|B(d - 2r - 1, n-k)|} + 1.$$

In the systematic/linear case the Zinoviev-Litsyn-Laihonen bound becomes:

$$|B(r, k)| \leq A_q(n - k, d - 2r).$$

Easy computations can be done in the case $d = 3$, since in this case $r$ can be at most $1$, so that: $|B(1, k)| = (q - 1)k + 1, A_q(n - k, d - 2r) = A_q(n - k, 1) = q^{n-k}$, $|B(1, n - k)| = (q - 1)(n - k) + 1$, $|B(d - 2r - 1, n - k)| = |B(0, n - k)| = 1$.

Our bound then reduces to: $0 \leq q^{n-k} - (q - 1)n - 1$, stronger then the Zinoviev-Litsyn-Laihonen bound, which, when $d = 3$, reduces to: $0 \leq q^{n-k} - (q - 1)k - 1$.

**Remark 1** Theorem 2 and Corollary 1 can be clearly generalized to any alphabet.
4 Experimental comparisons with other upper bounds

We have analyzed the case of linear codes, implementing Bound $B$. The algorithm to compute the bound takes as inputs $n, d$, and returns the largest $k$ (checks are done until $k = n - d + 1$) such that the inequality of the bound holds. If the inequality always holds in this range, $n - d + 1$ is returned. Then we compared our upper bound on $k$ with other bounds, restricting those which hold in the general non-linear case to the systematic case. In particular they give a bound on $A_q(n, d)$ instead of a bound on $k$. As a consequence, for example, if the Johnson bound returns the value $A_q(n, d)$ for a certain pair $(n, d)$, then we compare our bound with the value $\lfloor \log_q(A_q(n, d)) \rfloor$, which is the largest power $s$ of $q$ such that $q^s \leq A_q(n, d)$.

The inequality in Theorem 1 involves the value $A_q(n - k, d - 2r)$, which is the maximum number of words that we can have in a non-linear code of length $n - k$ and distance $d - 2r$. To implement Bound $B$ it is necessary to compute $A_q(n - k, d - 2r)$; when this value is unknown (we use known values only in the binary case for $n = 3, \ldots, 28$, $d = 3, \ldots, 16$), we return instead an upper bound on it, choosing the best between the Hamming (Sphere Packing), Singleton, Johnson, and Elias bound (the Plotkin bound is used when possible). Even though it is a very strong bound, we do not use the Levenshtein bound because it is very slow as $n$ grows. This means that if better values of $A_q(n - k, d - 2r)$ can be found, then Bound $B$ could return even tighter results.

Table 1 and 2 show a comparison between all bounds’ performances, except for Plotkin’s, due to its restricted range. For each bound and for each $q$ in the range $\{2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, 19, 23, 25, 27, 29\}$ we have computed, for all values $n = 3, \ldots, 100$ and $d = 3, \ldots, n - 1$, the percentage of cases the bound is the “best” known bound between Bound $B$, the Griesmer, Johnson, Levenshtein, Elias, Hamming and Singleton bound. Both wins and draws are counted in the percentage, since more than one bound may reach the best known bound, and in this case we increased the percentage of each best bound. Up to $q = 7$ the Levenshtein bound is the most performing. From $9 \leq q \leq 29$ we have that Bound $B$ is the most performing bound, and in particular, in the case $q = 29$, it is the best known bound almost 91% of the times.

Table 3, instead, shows some cases (one per each $q = 7, \ldots, 29$) where Bound $B$ beats all other known bounds. This happens from $q = 7$, for the range of $n$ considered. The letters B, J, H, G, E, S and L stands respectively for Bound $B$, Johnson, Hamming (Sphere Packing), Griesmer, Elias, Singleton, and Levenshtein bound. It can be seen that there are some cases where Bound $B$ is tight, as for the parameters $(9, 17, 7)$, for which there exists a code with distance 10.

Comparisons have been made using inner MAGMA ([MAG]) implementations of known upper bounds, except for the Johnson bound. For this bound we noted that the inner MAGMA implementation could be improved.

Acknowledgements

The first two authors would like to thank the third author (their supervisor). Partial results appear in [Gue09]. The authors would like to thank: Ludo Tolhuizen (Philips Group Innovation, Research), the MAGMA group and in particular John Cannon.
References


The following tables show the results computed in the range $n = 3, \ldots, 100$, $d = 3, \ldots, n - 1$.

### Table 1
When each bound is the best for $2 \leq q \leq 11$.

<table>
<thead>
<tr>
<th>$q$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bound B</td>
<td>38.02</td>
<td>31.20</td>
<td>31.20</td>
<td>31.94</td>
<td>40.73</td>
<td>48.64</td>
<td><strong>55.27</strong></td>
<td><strong>66.44</strong></td>
</tr>
<tr>
<td>Johnson</td>
<td>40.65</td>
<td>31.18</td>
<td>33.50</td>
<td>35.13</td>
<td>35.70</td>
<td>35.51</td>
<td>35.09</td>
<td>33.26</td>
</tr>
<tr>
<td>Hamming</td>
<td>18.12</td>
<td>15.65</td>
<td>16.37</td>
<td>16.35</td>
<td>16.03</td>
<td>15.88</td>
<td>15.57</td>
<td>14.69</td>
</tr>
<tr>
<td>Griesmer</td>
<td>56.32</td>
<td>39.83</td>
<td>32.32</td>
<td>29.14</td>
<td>30.91</td>
<td>36.97</td>
<td>43.28</td>
<td>55.15</td>
</tr>
<tr>
<td>Levenshtein</td>
<td><strong>72.65</strong></td>
<td><strong>69.68</strong></td>
<td><strong>66.27</strong></td>
<td><strong>64.02</strong></td>
<td><strong>60.80</strong></td>
<td><strong>58.24</strong></td>
<td>54.47</td>
<td>46.26</td>
</tr>
<tr>
<td>Elias</td>
<td>6.859</td>
<td>32.28</td>
<td>38.27</td>
<td>40.02</td>
<td>40.82</td>
<td>40.14</td>
<td>37.24</td>
<td>31.57</td>
</tr>
<tr>
<td>Singleton</td>
<td>0.000</td>
<td>0.021</td>
<td>0.084</td>
<td>0.189</td>
<td>0.610</td>
<td>0.926</td>
<td>1.241</td>
<td>3.619</td>
</tr>
</tbody>
</table>

### Table 2
When each bound is the best for $13 \leq q \leq 29$.

<table>
<thead>
<tr>
<th>$q$</th>
<th>13</th>
<th>16</th>
<th>17</th>
<th>19</th>
<th>23</th>
<th>25</th>
<th>27</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bound B</td>
<td><strong>76.43</strong></td>
<td><strong>81.61</strong></td>
<td><strong>82.75</strong></td>
<td><strong>85.42</strong></td>
<td><strong>88.11</strong></td>
<td><strong>88.72</strong></td>
<td><strong>89.40</strong></td>
<td><strong>90.77</strong></td>
</tr>
<tr>
<td>Johnson</td>
<td>30.80</td>
<td>26.61</td>
<td>24.87</td>
<td>21.88</td>
<td>17.08</td>
<td>15.51</td>
<td>14.37</td>
<td>13.34</td>
</tr>
<tr>
<td>Griesmer</td>
<td>63.39</td>
<td>71.91</td>
<td>72.27</td>
<td>71.94</td>
<td>69.79</td>
<td>69.43</td>
<td>68.65</td>
<td>67.87</td>
</tr>
<tr>
<td>Levenshtein</td>
<td>39.93</td>
<td>32.86</td>
<td>30.65</td>
<td>27.50</td>
<td>22.62</td>
<td>20.70</td>
<td>19.44</td>
<td>18.37</td>
</tr>
<tr>
<td>Elias</td>
<td>27.06</td>
<td>21.84</td>
<td>20.01</td>
<td>17.59</td>
<td>14.51</td>
<td>12.01</td>
<td>11.36</td>
<td>10.01</td>
</tr>
<tr>
<td>Singleton</td>
<td>4.439</td>
<td>4.629</td>
<td>6.985</td>
<td>6.712</td>
<td>10.08</td>
<td>12.01</td>
<td>14.12</td>
<td>18.01</td>
</tr>
</tbody>
</table>

### Table 3
Some cases where Bound B beats all the other bounds in the range $7 \leq q \leq 29$.

<table>
<thead>
<tr>
<th>$q$</th>
<th>n</th>
<th>d</th>
<th>B</th>
<th>J</th>
<th>H</th>
<th>G</th>
<th>E</th>
<th>S</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>45</td>
<td>24</td>
<td><strong>22</strong></td>
<td>21</td>
<td>24</td>
<td>23</td>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>51</td>
<td>24</td>
<td><strong>25</strong></td>
<td>27</td>
<td>28</td>
<td>26</td>
<td>28</td>
<td>28</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>7</td>
<td><strong>10</strong></td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>90</td>
<td>55</td>
<td><strong>30</strong></td>
<td>41</td>
<td>42</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
<td>9</td>
<td><strong>23</strong></td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>16</td>
<td>52</td>
<td>14</td>
<td><strong>38</strong></td>
<td>39</td>
<td>40</td>
<td>39</td>
<td>39</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>17</td>
<td>38</td>
<td>9</td>
<td><strong>29</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
<td>9</td>
<td><strong>33</strong></td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>34</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>23</td>
<td>91</td>
<td>17</td>
<td><strong>74</strong></td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>78</td>
</tr>
<tr>
<td>25</td>
<td>31</td>
<td>5</td>
<td><strong>28</strong></td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>27</td>
<td>88</td>
<td>24</td>
<td><strong>64</strong></td>
<td>66</td>
<td>67</td>
<td>65</td>
<td>66</td>
<td>65</td>
<td>69</td>
</tr>
<tr>
<td>29</td>
<td>100</td>
<td>29</td>
<td><strong>71</strong></td>
<td>74</td>
<td>74</td>
<td>72</td>
<td>74</td>
<td>72</td>
<td>76</td>
</tr>
</tbody>
</table>