Noname manuscript No.
(will be inserted by the editor)

A New Algorithm for Finding Low-Weight Polynomial Multiples and its Application to TCHo

Thomas Johansson · Carl Löndahl

the date of receipt and acceptance should be inserted later

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

Dept. of Electrical and Information Technology,
Lund University, P.O. Box 118, 221 00 Lund, Sweden
E-mail: {thomas, carl}@eit.lth.se

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 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 explicity 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* $\mathbf{v} = \begin{bmatrix} v_0 & v_1 & \cdots & v_{n-1} \end{bmatrix}$ *is the number of non-zero elements, i.e.,* $w_H(\mathbf{v}) = |\{i \ : \ v_i \neq 0, \ 0 \leq i < n\}|$*, while the weight of a polynomial* $P(x) = p_0 + p_1 x + \cdots + p_{n-1}x^{n-1}$ *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 (LWPMa)
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 (LWPMb)
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 $w = 2j$ (and $w = 2j-1$). The initial step consists of creating a list that contains the $\binom{n}{j}$ residues of the form $x^{i_1} + x^{i_2} + \cdots + x^{i_j} \bmod P(x)$, for $0 \le i_1 < i_2 < \cdots < i_j < n$. 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 $2j$ 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 $n^{\lceil (w-1)/4 \rceil}$ 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 $\mathcal{C}$. An (n, k) linear code is a linear subspace of dimension k of a vector space $\mathbb{F}_q^n$, defined by $\mathcal{C} \overset{\text{def}}{=} \{\mathbf{uG} \ : \ \mathbf{u} \in \mathbb{F}_q^k\}$ where $\mathbf{G}$ is a $k \times n$ matrix over $\mathbb{F}_q$, called the *generator matrix*. The problem is to find a vector $\begin{bmatrix} v_0 \ v_1 \ \cdots \ v_{n-1} \end{bmatrix} = \mathbf{v} \in \mathcal{C} \setminus \{\mathbf{0}\}$ such that $\mathbf{v}$ has a low weight w. This problem is commonly known as the function problem version of Subspace Weight and it is $\mathcal{NP}$-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

$$\mathbf{u} = \begin{bmatrix} u_0 \ u_1 \ \cdots \ u_{d-d_P} \end{bmatrix}$$

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 $\mathbf{u}$ such that $\mathbf{uG}(x)$ has exactly w non-zero coefficients, where

$$\mathbf{G}(x) = \begin{bmatrix} P(x) \\ xP(x) \\ \vdots \\ x^{d-d_P} P(x) \end{bmatrix}.$$

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

$$\mathbf{G} = \begin{bmatrix} p_0 & p_1 & \cdots & p_{d_P} & & & \\ & p_0 & p_1 & \cdots & p_{d_P} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & p_0 & p_1 & \cdots & p_{d_P} \end{bmatrix}, \qquad (1)$$

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 $\mathbf{G}$, 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 $\mathbf{G}$ 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 $xK(x), x^2K(x)\cdots, x^yK(x)$ 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:

$$\mathbf{G}_y = \begin{bmatrix} p_0 & p_1 & \cdots & p_{d_P} & & & & & \\ & p_0 & p_1 & \cdots & p_{d_P} & & & & \\ & & \ddots & \ddots & & \ddots & & & \\ & & & p_0 & p_1 & \cdots & p_{d_P} & & \\ & & & & p_0 & p_1 & \cdots & p_{d_P} & \\ & & & & & \ddots & \ddots & & \ddots \\ & & & & & & p_0 & p_1 & \cdots & p_{d_P} \end{bmatrix}. \qquad (2)$$

The gray rectangle represents $\mathbf{G}$ and everything outside represents the expansion. Let the expanded matrix in (2) be denoted $\mathbf{G}_y$ and the code it generates be denoted $\mathcal{C}_y$.

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 $\mathbf{\Gamma}$ that transforms the code $\mathcal{C}_y$ into a new code given by $\mathbf{G}_y\mathbf{\Gamma}$, 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 $\mathbf{G}_y$, i.e., adding the $(d+1)$th column to the first column of $\mathbf{G}_y$ and then removing the $(d+1)$th column from $\mathbf{G}_y$, 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 yth 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 $\mathbf{\Gamma}$, giving the new generator matrix $\mathbf{G}_y\mathbf{\Gamma}$. $\square$

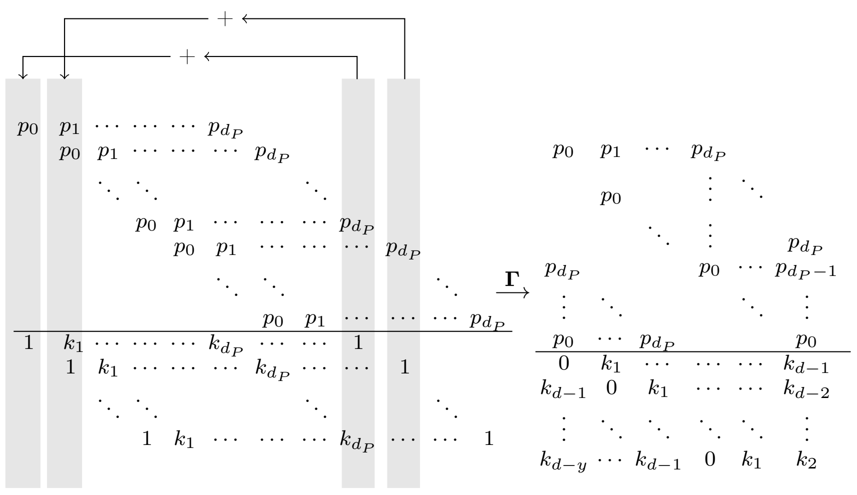

Fig. 1 The upper part above the line illustrates how the columns in $\mathbf{G}_y$ are added to form $\mathbf{G}_y\boldsymbol{\Gamma}$. The lower part below the line shows how the weight w codewords are transformed into weight $w-2$ codewords.

The matrix product $\mathbf{G}_y\boldsymbol{\Gamma}$ 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 $\mathbf{G}_y\boldsymbol{\Gamma}$, 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 $\mathcal{C}$, assuming that exactly one such codeword exists.

Theorem 2 (Johansson & Löndahl)

$$C^*(n,k,w)=\min_{p,z,l}\frac{\binom{n}{w}\left((n-k)^2(n+k)/2+\binom{k}{p/2}pz/2+\binom{k}{p}2^{-z}pl+\left(\binom{k}{p}2^{-z}\right)^2 2^{-l}2p(n-k)\right)}{\binom{k}{2p}\binom{n-k-z-l}{w-2p}\left(1-(1-2^{-z})^{\binom{2p}{p}/2}\right)},$$

where p, z and l are algorithm parameters.

Proof See [16] for a detailed proof. □

For an expanded (n,k) code $\mathcal{C}_y$ (according to (2)) with weight reduction by $\boldsymbol{\Gamma}$, we have $(n,k)=(d+1,d-d_P+1+y)$. Running Algorithm 2 on $\mathbf{G}_y\boldsymbol{\Gamma}$ will require complexity according to the following theorem.

Algorithm 1 Improved Stern

```
Input: Generator matrix $\mathbf{G}$, parameters $p, z, l$
Output: A codeword of weight $w$

1. Let $\mathbf{G}' = \pi(\mathbf{G})$, where $\pi$ is a random column permutation and $\mathbf{G}$ is the generator
   matrix with $k$ rows and $n$ columns.
2. Bring the generator matrix $\mathbf{G}'$ to systematic form: $[\mathbf{I}\ \mathbf{Z}\ \mathbf{L}\ \mathbf{J}]$, where $\mathbf{I}$ is the $k \times k$
   identity matrix, $\mathbf{Z}$ 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 $\phi^{\mathbf{Z}}(\mathbf{x})$ be a linear map $\phi^{\mathbf{Z}} : \mathbb{F}_2^n \to \mathbb{F}_2^z$. With $\mathbf{x} = [x_0\ x_1\ \cdots\ x_{n-1}]$,
   then $\phi(\mathbf{x}) = [x_k\ x_{k+1}\ \cdots\ x_{k+z-1}]$. Likewise, we define $\phi^{\mathbf{L}} : \mathbb{F}_2^n \to \mathbb{F}_2^l$ such that
   with $\phi(\mathbf{x}) = [x_{k+z}\ x_{k+z+1}\ \cdots\ x_{k+z+l-1}]$.
3. Let $\mathbf{u}$ run through all weight $p$ vectors of length $k$. Store all vectors $\mathbf{x} = \mathbf{u}\mathbf{G}'$
   such that $\phi^{\mathbf{Z}}(\mathbf{x}) = [0\ 0\ \cdots\ 0]$ in a sorted list $H_1$, sorted according to $\phi^{\mathbf{L}}(\mathbf{x})$.
   This is done by constructing a list $H_0$ containing all vectors $\mathbf{x} = \mathbf{u}\mathbf{G}'$ where $\mathbf{u}$
   runs through all weight $p/2$ vectors. Then add all pairs of vectors $\mathbf{x}, \mathbf{x}' \in H_0$
   in the list with $\phi^{\mathbf{Z}}(\mathbf{x}) = \phi^{\mathbf{Z}}(\mathbf{x}')$ and such that the largest index of the nonzero
   entries in $\mathbf{x}$ is smaller than the smallest index of nonzero entries in $\mathbf{x}'$.
4. As in previous step, combine the list $H_1$ with itself to receive a new list $H_2$ of all
   codewords $\mathbf{x} = \mathbf{u}\mathbf{G}'$ with $\mathbf{u}$ of weight $2p$, such that $\phi^{\mathbf{L}}(\mathbf{x}) = [0\ 0\ \cdots\ 0]$.
5. For each $\mathbf{x} \in H_2$, check if the weight of $\mathbf{x}$ 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 $\mathbf{G}_y\mathbf{\Gamma}$, in agreement with Theorem 1.
4. Input $\mathbf{G}_y\mathbf{\Gamma}$ into Algorithm 1, using optimum parameters with respect to (3), to
   find one codeword $\mathbf{u}$ among the $y + 1$ weight $w$ codewords that represent $K(x)$.
5. From $\mathbf{u}$, 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*

$$\min_{y \geq 0} \frac{C^*(d + 1, d - d_P + 1 + y, w - 2)}{y + 1} \tag{3}$$

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 ξ 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

$$\mathbf{K} = \begin{bmatrix} 1 & k_1 & k_2 & \cdots & \cdots & \cdots & k_{d-1} & 1 & & & \\ & 1 & k_1 & k_2 & \cdots & \cdots & \cdots & k_{d-1} & 1 & & \\ & & \ddots & \ddots & \ddots & & & & \ddots & \ddots & \\ & & & 1 & k_1 & k_2 & \cdots & \cdots & \cdots & k_{d-1} & 1 \end{bmatrix}.$$

The weight reduction of $\mathbf{G}_y$ by $\mathbf{\Gamma}$, will result in a new matrix

$$\mathbf{K\Gamma} = \begin{bmatrix} 0 & k_1 & k_2 & \cdots & \cdots & \cdots & \cdots & k_{d-1} \\ k_{d-1} & 0 & k_1 & k_2 & \cdots & \cdots & \cdots & k_{d-2} \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ k_{d-y} & \cdots & k_{d-1} & 0 & k_1 & k_2 & \cdots & k_{d-y-1} \end{bmatrix}.$$

Permutation of $\mathbf{G}_y\mathbf{\Gamma}$ permutes all codewords accordingly, thus permuting the matrix $\mathbf{K}$ as well. So we have that

$$\pi(\mathbf{K\Gamma}) = \left[\begin{array}{cccc|c} k_{i_1} & k_{i_2} & \cdots & k_{i_{d_P}} & \\ k_{i_1-1} & k_{i_2-1} & \cdots & k_{i_k-1} & \\ \vdots & \vdots & & \vdots & \cdots \\ k_{i_1-y} & k_{i_2-y} & \cdots & k_{i_{d_P}-y} & \end{array}\right].$$

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 $\pi(\mathbf{K\Gamma})$ 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 conlude 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 $\mathcal{O}\left(\xi^{-1}\right)$, this concludes the proof. □

4.1 Simulation results

We consider a toy example of LWPMb, running Algorithm 2 on an instance with a polynomial

$$\begin{aligned} P(x) = \; & 1 + x^1 + x^3 + x^6 + x^7 + x^8 + x^{10} + x^{11} + x^{16} + x^{20} + \\ & x^{25} + x^{28} + x^{29} + x^{32} + x^{33} + x^{34} + x^{37} + x^{38} + x^{39}. \end{aligned}$$

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

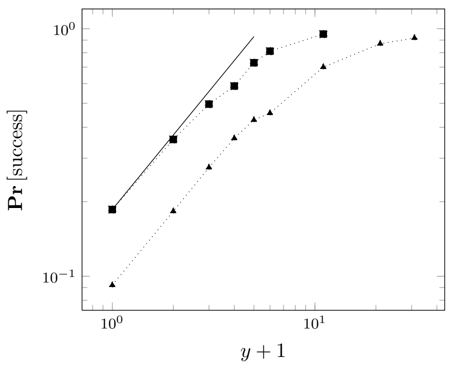

Fig. 2 The probability of success in each iteration as a function of codeword multiplicity.

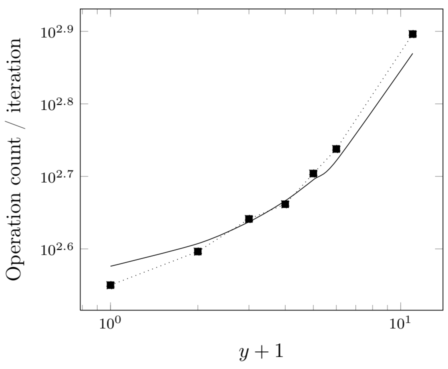

Fig. 3 Operation count in each iteration as a function of codeword multiplicity.

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 $\mathbf{G}_{\text{rep}}$ be a generator matrix of a repetition code of length l. The plaintext $\mathbf{m} \in \mathbb{F}_2^{128}$ is repeated and the result is truncated to fit a length l. These two steps can be represented by multiplication with $\mathbf{G}_{\text{rep}}$.

In encoding, let $\mathbf{r} = \begin{bmatrix} r_0 \; r_1 \; \cdots \; r_{l-1} \end{bmatrix}$ be a random string of l independent bits with bias $\mathbf{Pr}\left[r_i = 0\right] = \frac{1}{2}(1+\gamma)$ (called γ-biased). Here, $\mathbf{r}$ 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 $\mathbf{p}$ is generated and truncated to l bits. The encryption is done by adding the three vectors to form the ciphertext $\mathbf{c} \in \mathbb{F}_q^l$, where $\mathbf{c} = \mathbf{m}\mathbf{G}_{\text{rep}} + \mathbf{r} + \mathbf{p}$.

In encoding, given the secret low-weight polynomial $K(x)$ of fixed degree, we can construct the matrix

$$\mathbf{M} = \begin{bmatrix} k_0 & k_1 & \cdots & k_{d_K} & & & \\ & k_0 & k_1 & \cdots & k_{d_K} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & k_0 & k_1 & \cdots & k_{d_K} \end{bmatrix}.$$

For the decoding step, consider the product $\mathbf{t} = \mathbf{c}\mathbf{M}^{\mathrm{T}}$. From the encryption step we have that

$$\mathbf{t} = (\mathbf{m}\mathbf{G}_{\mathrm{rep}} + \mathbf{r} + \mathbf{p})\mathbf{M}^{\mathrm{T}} = \mathbf{m}\mathbf{G}_{\mathrm{rep}}\mathbf{M}^{\mathrm{T}} + \mathbf{r}\mathbf{M}^{\mathrm{T}} + \mathbf{p}\mathbf{M}^{\mathrm{T}} = \mathbf{m}\mathbf{G}_{\mathrm{rep}}\mathbf{M}^{\mathrm{T}} + \mathbf{r}\mathbf{M}^{\mathrm{T}}.$$

Since $P(x)$ divides $K(x)$, we have that $\mathbf{p}\mathbf{M}^{\mathrm{T}} = \mathbf{0}$. Recall that each bit element in $\mathbf{r}$ was γ-biased. $K(x)$ has weight w and consequently, each element in $\mathbf{r}$ will be a sum of w variables that each have bias γ. Therefore, each element in $\mathbf{r}\mathbf{M}^{\mathrm{T}}$ will be γ^w-biased. Here, majority decision decoding can be used to decode

$$\mathbf{t} = \mathbf{m}\left(\mathbf{G}_{\mathrm{rep}}\mathbf{M}^{\mathrm{T}}\right) + \mathbf{r}\mathbf{M}^{\mathrm{T}},$$

i.e., to find a solution $\mathbf{m}$ such that the residual $\mathbf{r}\mathbf{M}^{\mathrm{T}}$ 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 LWPMB 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 1 The first complexity column gives the complexity of using Stern's algorithm, the second column for using Algorithm 1 and 2, respectively. The optimum column gives the optimal choice of y in Algorithm 2.

Instance parameters			Complexity			Optimum
d_K	d_P	w	Stern	Algorithm 1	Algorithm 2	y_{opt}
25820	7000	45	100.69	100.08	90.61	200
24730	12470	67	85.56	85.75	77.65	230
44677	4433	25	97.48	96.47	84.15	250
24500	8000	51	98.91	98.45	89.48	200
17600	8795	81	97.43	96.51	89.21	110
31500	13200	65	99.84	99.80	91.13	250

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* $\mathbf{G}_y\boldsymbol{\Gamma}$ *now has dimension* 40495×44678 *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 LWPMB. 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

1. M. Ågren, M. Hell, T. Johansson and C. Löndahl Improved Message Passing Techniques in Fast Correlation Attacks on Stream Ciphers. In *7th International Symposium on Turbo Codes & Iterative Information Processing*, 2012.
2. L. El Aimani and J. von zur Gathen. Finding low weight polynomial multiples using lattices. In *Cryptology ePrint Archive*, volume Report 2007/423, 2007.
3. J. Aumasson, M. Finiasz, W. Meier, and S. Vaudenay. TCHo : A hardware-oriented trapdoor cipher. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *ACISP*, volume 4586 of *Lecture Notes in Computer Science*, pages 184–199. Springer-Verlag, 2007.
4. A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^n/20$: How $1 + 1 = 0$ improves information set decoding. In D. Pointcheval, T. Johansson, editors, *Advances in Cryptology—EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer-Verlag, 2012.
5. D. J. Bernstein. Introduction to post-quantum cryptography. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 1–14. Springer-Verlag, 2009.
6. D. J. Bernstein, T. Lange and C. Peters Smaller decoding exponents: Ball collision decoding. In P. Rogway, editor, *Advances in Cryptology—CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer-Verlag, 2011.
7. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44:367–378, 1998.

8. A. Canteaut and M. Trabbia. Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5. In B. Preneel, editor, *Advances in Cryptology—CRYPTO 2000*, volume 2442 of *Lecture Notes in Computer Science*, pages 573–588. Springer-Verlag, 2000.
9. P. Chose, A. Joux, and M. Mitton. Fast correlation attacks: An algorithmic point of view. *Lecture Notes in Computer Science*, 2332:209–221, 2002.
10. F. Didier and Y. Laigle-Chapuy. Finding low-weight polynomial multiples using discrete logarithm. In A. Goldsmith, A. Shokrollahi, M. Medard, and R. Zamir, editors, *International Symposium on Information Theory—ISIT 2007*. IEEE, CCSd, 2007.
11. M. Finiasz and S. Vaudenay. When stream cipher analysis meets public-key cryptography. In E. Biham and A. M. Youssef, editors, *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 266–284. Springer-Verlag, 2006.
12. M. Finiasz and N. Sendrier Security Bounds for the Design of Code-Based Cryptosystems. In M. Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009*, volume 4586 of *Lecture Notes in Computer Science*, pages 88–105. Springer-Verlag, 2009.
13. J. D. Golić. Computation of low-weight parity-check polynomials. *Electronic Letters*, 32(21):1981–1982, October 1996.
14. M. Herrmann and G. Leander. A practical key recovery attack on basic TCHo. In S. Jarecki and G. Tsudik, editors, *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 411–424. Springer-Verlag, 2009.
15. J. Stern. A method for finding codewords of small weight. In J. Wolfmann and G. D. Cohen, editors, *Coding theory and applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer-Verlag, 1989.
16. T. Johansson and C. Löndahl. An improvement to Stern's algorithm, internal report, 2011 `http://lup.lub.lu.se/record/2204753`
17. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}\left(2^{0.054n}\right)$. In D.-H. Lee and X. Wang, editors, *Advances in Cryptology—ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer-Verlag, 2011.
18. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report 42–44*, pages 114–116, 1978.
19. W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.
20. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, 20-22 November 1994, Santa Fe, New Mexico, USA*, pages 124–134. IEEE Press, 1994.
21. D. Wagner. A generalized birthday problem. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer-Verlag, 2002.