ECC i akademia vs. industrien

Øyvind Grinde

Conax AS

2007

Introduksjon

Kampen mellom ECC og RSA har pågått lenge. I akademia går ECC av som vinner, mens i industrien er det fortsatt RSA som gjelder.

RSA

- I 1977 publiserte Rivest, Shamir og Adleman algoritmen RSA som gjør det mulig med asymmetrisk kryptografi.
- Det tok lang tid før algoritmen fikk stor industriell utbredelse.
- I dag brukes algoritmen for tilnærmet all sikkerhet på verdensveven.

ECC

- Elliptiske kurver har vært studert i lang tid, men i 1985 presenterte Neal Koblitz and Victor S. Miller (uavhengig av hverandre) hvordan elliptiske kurver kunne brukes for kryptografi (ECC).
- De viste hvordan ECC er bedre enn RSA med hensyn både på effektiv utregning og lengden på nøklene.

Utbredelse

- Akademia anbefaler ECC på det sterkeste, men algoritmen er fremdeles lite utbredt i industrien.
- Vi skal se litt på hva som kan være årsaken til dette.

Kryptografi

Vi skal kort oppfriske kryptografien for RSA og ECC, samt se på de mest bruke standarder og protokoller.

RSA

Modulus n er produktet av to store primtall p og q.

Et par (e, d) er valgt slik at $e \equiv d \pmod{\phi(n)}$.

Offentlig nøkkel: (e, n)
Privat nøkkel: (d, n).

Kryptering: $c = (m^e \bmod n)$
Dekryptering: $m = (c^d \bmod n)$.
Signering: $sig = (hash^d \bmod n)$

Styrken til RSA baserer seg på hvor vanskelig det er å faktorisere et tall som er produktet av to store primtall.

ECC

En elliptisk kurve E over en kropp K er definert ved følgende ligning:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Et punkt P på kurven er et par (x, y) med $x, y \in K$ som tilfredstiller ligningen. ECC består av en kurve E og et punkt P på kurven.

Privat nøkkel: $d \in \mathbf{Z}$.
Offentlig nøkkel: Q gitt ved $Q = dP$ $(dP = P + P + \ldots + P)$

Styrken til ECC baserer seg på det diskrete logaritmeproblemet for elliptiske kurver, som er å finne d, gitt Q og P, der $Q = dP$.

Protokoller

- PKCS #1 v1.5 og v2.1
- X.509 (ITU)
- X9.31-1988 (American National Standard for Financial Services)
- DSS (FIPS)
- ECDSA (ANSI X9.62, FIPS 186-2, IEEE 1363-2000 og ISO/IEC 15946-2)
- EC-KCDSA (ISO/IEC 15946-2)
- ECIES (ANSI X9.63, ISO/IEC 15946-3 og IEEE P1363a draft)
- PSEC (ISO 18033-2 draft)

Industri

Vi skal kort se på hvem som bruker RSA og ECC, hva de har til felles og hvorfor de velger som de gjør.

Bruk

- Bank
- Mobil
- Betal TV
- Adgangskontroll (Bygg, bil, pc, etc...)
- WEB (SSL og HTTPS)
- e-post (PGP og X.509)

Fellestrekk

Eier begge endene av kommunikasjon

Både sender og mottaker i beskyttet hardware.

Begrensninger

- Kringkasting
- Begrenset båndbredde
- Flere brukere må dekryptere samme melding.
  - Unike meldinger til for eksempel 10 millioner kunder...
  - "Broadcast encryption" bruker ofte ikke standardiserte algoritmer...
- Korte meldinger
  - Smartkort grensesnitt (ISO 7816-3) - 256 bytes
  - DVB Transportpakke - 188/255 bytes
  - Tekstmelding - 160 tegn (7 bits)

Egenskaper

Kryptografi er en nødvendig ulempe som:

- gjør implementasjonen mer komplisert
- vanskeliggjør testing
- stjeler båndbredde

Vi skal se på nøkler, signaturer og krypterting.

Nøkkellengde

1024 bits RSA tilsvarer 160 bits ECC. Dette er ansett som sikkert frem til år 2010 (NIST).

2048 bits RSA tilsvarer 224 bits ECC. Dette er ansett som sikkert i lang tid fremover.

RSA: 2048 (256 bytes) - med modulus 512 bytes.
ECC: 224 (28 bytes) - med kurve 150 - 250 bytes.

Når vi oppgir størrelser senere i presentatsjonen baserer dette seg på 2048 / 224 bits sikkerhet, selv om det vil ta lang tid før disse størrelsene er fullt ut støttet av industrien.

Signaturer

En RSA signatur av en melding har normalt samme lengde som RSA nøkkelen. Det vil si 256 bytes for en 2048 bits RSA nøkkel. (e.g. PKCS #1 og X9.31-1988)

En ECC signatur av en melding har normalt dobbelt lengde av ECC nøkkelen. Det vil si 56 bytes for en 224 bits ECC nøkkel. (e.g. ECDSA og EC-KCDSA)

Kryptering - RSA

For RSA kryptering er PKCS #1 v1.5 mest utbredt. Maksimal lengde for meldingen som krypteres er $k - 11$, hvor k er lengden av modulus i hele bytes. Det vil si at kryptering koster 11 bytes per 255 bytes melding (4,3%).

PKCS #1 v2.1 bruker hash av meldingen for integritet. Det koster 22 bytes (8,6%) for SHA-1 (hash lengde + 2)

Kryptering - ECC

ECIES brukes for å sende en symmetrisk nøkkel som meldingen er kryptert med. Kryptoteksten består av (R, C, t), hvor:

- R - punkt på kurven (28 bytes)
- C - symmetrisk kryptering
  - AES/CBC/CTS eller AES/CTR (16 bytes ekstra)
  - AES/ECB/CTS (0 bytes ekstra)
- t - MAC (e.g. HMAC på 20 bytes)

Det vil si at kryptering med ECC har en "overhead" på 64 bytes (25%)!

ECC er i tillegg sårbar for strømtrekksanalyse både under asymmetrisk og symmetrisk dekryptering.

Når standardene ikke holder

Vi har sett at meldingene vi ønsker å beskytte i mange tilfeller er begrenset til maksimalt 256 bytes eller mindre.

Videre har vi sett at standardene har en "overhead" som tilsvarer nesten hele vår maksimale meldingslengde.

Hvilke muligheter har vi da?

Løsning for RSA

La en "trusted third party" (TTP) dele ut nøklene. Partene som skal kommunisere får modulus og en eksponent hver. (Begge eksponenter holdes hemmelig)

For PKCS #1 v2.1 vil hashverdien fungere som en signatur. Mottager dekrypterer, leser melding, og verifiserer hash.

Authoriserte mottagere er de eneste som kan dekryptere og lese meldingen. Mottager kan videre verifisere at meldingen stammer fra riktig avsender.

Avslutning

- ECC har stor interesse i akademia fordi det er et interessant forskningsfelt.
- ECC har sin fordel på grunn av korte nøkler, men standardene er laget for lange meldinger, og da er ikke nøkkellengde av stor betydning.
- ECC er mer komplisert. (e.g. RSA byte for byte)

ECC utfordring

Finnes det en god løsning for ECC kryptering av korte meldinger?

Vi ønsker å se gode protokoller for ECC kryptering av korte meldinger.