Elliptic Curve Hash (and Sign)

ECOH (and the 1-up problem for ECDSA)

Daniel R. L. Brown

Certicom Research

ECC 2008, Utrecht, Sep 22-24 2008

Outline

1. ECOH
   - Background
   - Evolution
   - Implementation
   - CFV
2. One-Up Problem for ECDSA
3. Conclusion

Elliptic Curve Only Hash

Definition (High level)

Pad message block M_i into a point P_i.

$$T = \sum_i P_i \tag{1}$$

Do the same for T. Truncate to get hash H.

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call "AHS"

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call "AHS"

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call “AHS”

certicom

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call "AHS"

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call "AHS"

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call “AHS”

Motivation: SHA-3

- Wang, Feng, Lai, Yu: collision FOUND in MD5.
- Wang, Yin, Yu: 2^{69} collision algorithm for SHA-1
- Wang, Yao, Yao: 2^{63} collision algorithm for SHA-1
- NIST: please use SHA-2
- NIST: is SHA-2 ok?
- NIST: SHA-3 competition, AES-style
- Some like to call “AHS”

Discrete Log Hash: CHP

Definition (Chaum, van Heijst, Pfitzmann (1991))

$H(m, n) = mP + nQ$

Theorem

A collision in H gives $\log_P(Q)$.

Proof.

If $H(a, b) = H(c, d)$, then

$$aP + bQ = cP + dQ \tag{2}$$

and solving $\log_P(Q) = \frac{a-c}{d-b} \bmod n$. □

Discrete Log Hash: CHP

Definition (Chaum, van Heijst, Pfitzmann (1991))

$H(m, n) = mP + nQ$

Theorem

A collision in H gives $\log_P(Q)$.

Proof.

If $H(a, b) = H(c, d)$, then

$$aP + bQ = cP + dQ \tag{2}$$

and solving $\log_P(Q) = \frac{a-c}{d-b} \bmod n$. □

Discrete Log Hash: CHP

Definition (Chaum, van Heijst, Pfitzmann (1991))

$H(m, n) = mP + nQ$

Theorem

A collision in H gives $\log_P(Q)$.

Proof.

If $H(a, b) = H(c, d)$, then

$$aP + bQ = cP + dQ \tag{2}$$

and solving $\log_P(Q) = \frac{a-c}{d-b} \bmod n$. □

CHP Pros and Cons

- Provably secure assuming ECDLP hard.
- $3m/2$ EC adds per $2m$ bits.
- Compression factor 2, must be iterated.

Discrete Log Hash 2: MuHASH

Definition (Bellare and Micciancio (1997))

Let $P_i = F(i \| M_i)$, where F is a "random oracle". Let

$$H = \sum_i P_i \tag{3}$$

MuHASH Advantages

- One EC add per m bits.
  - E.g. 384 times faster than CHP.
- Parallelizable.
- Incremental:
  - $H' = H - P_i + P'_i$
- Provably secure, assuming ECDLP hard and F random oracle.

MuHASH Disadvantages

- Assumes F is a random oracle.
- Insecure if F insecure.
  - Must already have a collision-resistant F.
  - SHA-1? SHA-2? SHA-3?

ECOH's Design Rationale

- Leverage from MuHASH:
  - Speed.
  - Parallelizability.
  - Incrementality.
- Avoid reliance on pre-existing F.

EECH

- Replace F by fixed key block cipher:

$$H = \sum_i F(i \| M_i) \tag{4}$$

- Encrypted Elliptic Curve Hash (EECH) born.
- No collisions in F, guaranteed.
- Model F by ideal cipher.
- Rehash Bellare and Micciancio's security proof.

EECH

- Replace F by fixed key block cipher:

$$H = \sum_i F(i \| M_i) \tag{4}$$

- Encrypted Elliptic Curve Hash (EECH) born.
- No collisions in F, guaranteed.
- Model F by ideal cipher.
- Rehash Bellare and Micciancio's security proof.

EECH

- Replace F by fixed key block cipher:

$$H = \sum_i F(i \| M_i) \tag{4}$$

- Encrypted Elliptic Curve Hash (EECH) born.
- No collisions in F, guaranteed.
- Model F by ideal cipher.
- Rehash Bellare and Micciancio's security proof.

certicom

EECH

- Replace F by fixed key block cipher:

$$H = \sum_i F(i \| M_i) \tag{4}$$

- Encrypted Elliptic Curve Hash (EECH) born.
- No collisions in F, guaranteed.
- Model F by ideal cipher.
- Rehash Bellare and Micciancio's security proof.

EECH

- Replace F by fixed key block cipher:

$$H = \sum_i F(i \| M_i) \tag{4}$$

- Encrypted Elliptic Curve Hash (EECH) born.
- No collisions in F, guaranteed.
- Model F by ideal cipher.
- Rehash Bellare and Micciancio's security proof.

Oops: Not 1-way

- Unlike MuHASH, F now invertible.
- If adversary knows M_1 and M_3 but not M_2, then

$$2\|M_2 = F^{-1}(H(M_1, M_2, M_3) - F(1\|M_1) - F(3\|M_3)) \tag{5}$$

Oops: Not 1-way

- Unlike MuHASH, F now invertible.
- If adversary knows M_1 and M_3 but not M_2, then

$$2\|M_2 = F^{-1}(H(M_1, M_2, M_3) - F(1\|M_1) - F(3\|M_3)) \qquad (5)$$

certicom

Fix it up.

- Post-process with one-way function?
  - Scalar multiply?
  - EECH again?
  - Pairing?
  - Checksum in extra block?
- Seems to thwart block inversion attack.
- Interferes with incrementality.

Ouch: Not collision resistant!

Let

$$2\|D = F^{-1}(F(1\|A) + F(2\|B) - F(1\|C)) \tag{6}$$

Probability of index 2 appearing depends its bit length. Try that many C values, until it works.

Then

$$F(1\|A) + F(2\|B) = F(1\|C) + F(2\|D), \tag{7}$$

i.e. a collision $H(A, B) = H(C, D)$.

Second preimage attack!

Fix it again.

- Pad M_i, before applying F.
- If F random enough, inverting will not give requisite padding.

ECOH

- Now that EECH is all fixed ...
- just set F to the identity function.
- Elliptic Curve Only Hash.

ECOH

- Now that EECH is all fixed ...
- just set F to the identity function.
- Elliptic Curve Only Hash.

certicom

ECOH

- Now that EECH is all fixed ...
- just set F to the identity function.
- Elliptic Curve Only Hash.

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

certicom

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

certicom

ECOH vs. EECH

- Purity of ECOH.
- No dependence on ideal cipher model.
- No performance cost of enciphering.
  - ECOH is already slow enough.
- Is it more crazy to:
  - encrypt with a fixed key,
  - do nothing?

ECOH Security Proof?

- Generic group model!
  - Detailed version in progress.
- Big deal ...

ECOH Security Attack!?!

- Semaev summation polynomial

$$f_n(X_1, \ldots, X_n) = 0$$

if and only if there exist Y_i with

$$(X_1, Y_1) + \cdots + (X_n, Y_n) = 0.$$

- Degree in each variable 2^{n-2}

Second Preimage Attack on ECOH

- Given X_3 and X_4.
- Find X_1 and X_2, such that

$$(X_1, Y_1) + (X_2, Y_2) = (X_3, Y_3) + (X_4, Y_4)$$

which implies

$$f_4(X_1, X_2, X_3, X_4) = 0$$

- Total degree $2(2^{4-2}) = 4$.
- $X_i = c_i Z_i + d_i$, where Z_i has low degree.

$$g(Z_1, Z_2) = 0$$

Security Proof??????

- Semaev: low degree solutions to Summation polynomials can be used to solve ECDLP.
- Contrapositive: if ECDLP hard, then hard to find low degree solutions.
- But: ECOH degrees much higher than Semaev degrees.

certicom

Security Proof??????

- Semaev: low degree solutions to Summation polynomials can be used to solve ECDLP.
- Contrapositive: if ECDLP hard, then hard to find low degree solutions.
- But: ECOH degrees much higher than Semaev degrees.

Security Proof??????

- Semaev: low degree solutions to Summation polynomials can be used to solve ECDLP.
- Contrapositive: if ECDLP hard, then hard to find low degree solutions.
- But: ECOH degrees much higher than Semaev degrees.

Curve Choice

- NIST recommended curves:
  - B-283,
  - B-409,
  - B-571.

Why Binary?

- y solved by quadratic equation involving x containing padded message block.
- Quadratic equations faster in binary fields than in prime fields
  - Use linear half-trace function (not square root)
  - Use look up tables.
- Bonus: Intel announced AVX will include binary polynomial multiplier.

Why Binary?

- y solved by quadratic equation involving x containing padded message block.
- Quadratic equations faster in binary fields than in prime fields
  - Use linear half-trace function (not square root)
  - Use look up tables.
- Bonus: Intel announced AVX will include binary polynomial multiplier.

Why Binary?

- y solved by quadratic equation involving x containing padded message block.
- Quadratic equations faster in binary fields than in prime fields
  - Use linear half-trace function (not square root)
  - Use look up tables.
- Bonus: Intel announced AVX will include binary polynomial multiplier.

Why Binary?

- y solved by quadratic equation involving x containing padded message block.
- Quadratic equations faster in binary fields than in prime fields
  - Use linear half-trace function (not square root)
  - Use look up tables.
- Bonus: Intel announced AVX will include binary polynomial multiplier.

Why Binary?

- y solved by quadratic equation involving x containing padded message block.
- Quadratic equations faster in binary fields than in prime fields
  - Use linear half-trace function (not square root)
  - Use look up tables.
- Bonus: Intel announced AVX will include binary polynomial multiplier.

Reference implementation

- Coded by Matt J. Campagna (who also helped with specification of ECOH details)
- Features:
  - Bit lookups for trace function
  - Table lookups for squaring and half-trace
  - Basic shift-and-xor polynomial multiply
  - Affine coordinates
- Rate on a desktop: 0.14 MB/s

Possible optimizations

- Other coordinates?
  - Not predicted to help.
- Better multiplication:
  - Should help somewhat.
- Simultaneous inversions:
  - Each solving for y requires inversion.
  - Each addition requires inversion.
  - These can be replaced a few inversion and a corresponding number of multiplies.
  - Predicted speedup: maybe five times?
- Parallelization

Hash with a Twist

- Bernstein: x-only DH with "invalid" x thrown to the twist.
- EECH/ECOH: every x maps to a point on curve or its twist
- Get one total and twisted total
- Sum these on curve over quadratic extension.

Dreaming doesn't hurt

0.14 MB/s
x 5 (simultaneous inversion, etc.)
x 10 (Intel AVX)
x 10 (ten CPU multicore)
=
70 MB/s
Faster than SHA-1?

People who have helped me

- Matt Campagna
- René Struik

Call for Volunteers

- Implementers
- Cryptanalysis
- Security provers

Convertible Group

Definition

A group G and a function $f : G \to \mathbb{Z}$.

- Use multiplicative notation for G.
- Call f the conversion function.

One-Up Problem

Definition

Given $a, b \in G$, find c such that

$$c = ab^{f(c)} \tag{8}$$

- One is up: a^1.
- One c is up.

Convertible DSA

Definition

Let $g \in G$ have order n. Let $h : \{0,1\}^* \to \mathbb{Z}$ be a hash function. Then (r, s) is a valid signature on message $m \in \{0,1\}^*$ under public key $y \in G$, only if $\gcd(s, n) = 1$ and

$$r = f\left(\left(g^{h(m)} y^r\right)^{1/s \bmod n}\right). \tag{9}$$

- Includes DSA.
- Includes ECDSA.

So what's up with this problem?

Theorem

If the one-up problem for (G, f) is solvable, then Convertible DSA for (G, f, g, h) is forgeable.

Hard up?

Conjecture

For the (G, f) in ECDSA, solving the 1-up problem costs about n group operations and converstions.

Up's enough?

Conjecture

Convertible DSA resists universal forgery against key-only attacks (UF-KOA) if

1. *Discrete logs hard in G.*
2. *One up hard in (G, f).*
3. *Hash h mod n is rarely zero.*

More powerful forgery attacks resisted if hash has further security properties (e.g. collision resistance).

Up over log?

- If discrete logs easy, ...
- Can one-up problem be hard?
- Maybe, if f ...
- is random oracle.

Up under log?

- In generic group model,
- If advesary gets access to one-up oracle, then
- Discrete logs still hard.

Semilog problem

Definition (ECC 2001, Advances in ECC)

A semilog of y is a pair (r, s) which would be valid signature under public key y if the message had hash equal to one.

Theorem (ECC 2001/Advances in ECC)

ECDSA resists UF-KOA if and only if semilog is hard and hash is rarely zero.

Semilog = Fork(Log, 1up)

Theorem

The semilog problem, with one component is fixed, is equivalent to

- *the discrete log problem if r is fixed.*
- *the 1-up problem if s is fixed.*

Diffie-Hellman Disguised as One-Up

- If $f(x) = \log_g(x)$, then
- One-up problem equivalent to DHP
- This f is impractial, so
- result is only theoretical.

One-Up as Obstacle

- Pointcheval and Stern couldn't prove ECDSA secure in random oracle model, assuming only hard log.
- Paillier and Vergnaud argued ECDSA couldn't be proved secure in the random oracle model, assuming hard log (unless one-more log problem was easy).
- Perhaps one-up problem was hidden obstacle.
- Not possible to prove ECDSA secure given only hard log, because one-up could be easy.
- In practice, though, one-up seems harder than log!

One-Up as Obstacle

- Pointcheval and Stern couldn't prove ECDSA secure in random oracle model, assuming only hard log.
- Paillier and Vergnaud argued ECDSA couldn't be proved secure in the random oracle model, assuming hard log (unless one-more log problem was easy).
- Perhaps one-up problem was hidden obstacle.
- Not possible to prove ECDSA secure given only hard log, because one-up could be easy.
- In practice, though, one-up seems harder than log!

One-Up as Obstacle

- Pointcheval and Stern couldn't prove ECDSA secure in random oracle model, assuming only hard log.
- Paillier and Vergnaud argued ECDSA couldn't be proved secure in the random oracle model, assuming hard log (unless one-more log problem was easy).
- Perhaps one-up problem was hidden obstacle.
- Not possible to prove ECDSA secure given only hard log, because one-up could be easy.
- In practice, though, one-up seems harder than log!

One-Up as Obstacle

- Pointcheval and Stern couldn't prove ECDSA secure in random oracle model, assuming only hard log.
- Paillier and Vergnaud argued ECDSA couldn't be proved secure in the random oracle model, assuming hard log (unless one-more log problem was easy).
- Perhaps one-up problem was hidden obstacle.
- Not possible to prove ECDSA secure given only hard log, because one-up could be easy.
- In practice, though, one-up seems harder than log!

One-Up as Obstacle

- Pointcheval and Stern couldn't prove ECDSA secure in random oracle model, assuming only hard log.
- Paillier and Vergnaud argued ECDSA couldn't be proved secure in the random oracle model, assuming hard log (unless one-more log problem was easy).
- Perhaps one-up problem was hidden obstacle.
- Not possible to prove ECDSA secure given only hard log, because one-up could be easy.
- In practice, though, one-up seems harder than log!

ECDSA with ECOH

- No bit twiddling — pure algebra.
- Use the same curve for both.

Conclusion

- ECC: not just for PKC and RNGs, anymore!
- ECOH: who needs need bit twiddling, now?
- ECDSA: One-up? Okay.