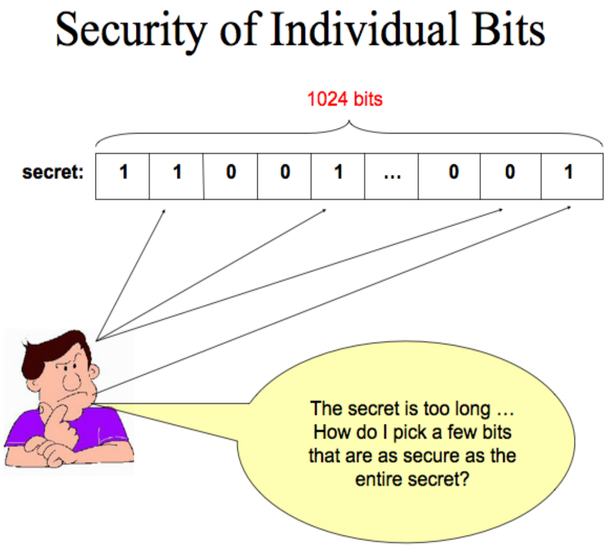
Hard-to-Compute Bits for Elliptic Curve-Based One-Way Functions

Alexandre Duc [1]   Dimitar Jetchev [1]

[1]EPFL, Switzerland

Crypto'2012, August 23rd, 2012, Santa Barbara, CA

Security of Individual Bits



- $E/\mathbb{F}_p$ - elliptic curve (p is a prime),

Pairing-based One-Way and FAPI-2

- $E/\mathbb{F}_p$ - elliptic curve (p is a prime),
- $\mathbb{G}$ - a large cyclic subgroup of points on E,

Pairing-based One-Way and FAPI-2

- $E/\mathbb{F}_p$ - elliptic curve (p is a prime),
- $\mathbb{G}$ - a large cyclic subgroup of points on E,
- $e\colon \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ - a cryptographic pairing,

Pairing-based One-Way and FAPI-2

- $E/\mathbb{F}_p$ - elliptic curve (p is a prime),
- $\mathbb{G}$ - a large cyclic subgroup of points on E,
- $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ - a cryptographic pairing,
- By fixing the second argument, one gets $f_Q\colon \mathbb{G} \to \mathbb{G}_T$,

$$f_Q(\bullet) = e(\bullet, Q)$$

Pairing-based One-Way and FAPI-2

- $E/\mathbb{F}_p$ - elliptic curve (p is a prime),
- $\mathbb{G}$ - a large cyclic subgroup of points on E,
- $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ - a cryptographic pairing,
- By fixing the second argument, one gets $f_Q\colon \mathbb{G} \to \mathbb{G}_T$,

$$f_Q(\bullet) = e(\bullet, Q)$$

- FAPI-2 problem is the problem of inverting this function

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

- Boneh–Franklin: identity-based encryption scheme

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

- Boneh–Franklin: identity-based encryption scheme
- Joux: three-party one-round key agreement protocol

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

- Boneh–Franklin: identity-based encryption scheme
- Joux: three-party one-round key agreement protocol
- Hess: identity-based signature scheme

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

- Boneh–Franklin: identity-based encryption scheme
- Joux: three-party one-round key agreement protocol
- Hess: identity-based signature scheme

FAPI-2 is Hard!

Solving FAPI-1 and FAPI-2 yields a solution to CDH.

Why is FAPI-2 relevant?

The security of the following schemes relies on the hardness of solving FAPI-2:

- Boneh–Franklin: identity-based encryption scheme
- Joux: three-party one-round key agreement protocol
- Hess: identity-based signature scheme

FAPI-2 is Hard!

Solving FAPI-1 and FAPI-2 yields a solution to CDH.

Our Contribution

Assuming the hardness of FAPI-2, we show that all the bits of the input to the pairing-based one-way function are secure.

(Short) Weierstrass Equations

Equations $E_{a,b} : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0.$

(Short) Weierstrass Equations

Equations $E_{a,b} : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0.$

Two Weierstrass equations might represent isomorphic curves.

(Short) Weierstrass Equations

Equations $E_{a,b} : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p,\ 4a^3 + 27b^2 \neq 0.$

Two Weierstrass equations might represent isomorphic curves.

Isomorphism classes

Two elliptic curves $E_{a,b}$ and $E_{a',b'}$ are isomorphic (over $\mathbb{F}_p$) **if and only if** $a' = \lambda^{-4}a,\ b' = \lambda^{-6}b$ for some $\lambda \in \mathbb{F}_p^\times$. The isomorphism between $E_{a,b}$ and $E_{a',b'}$ is given by

$$(x, y) \mapsto (\lambda^2 x, \lambda^3 y).$$

(Short) Weierstrass Equations

Equations $E_{a,b} : y^2 = x^3 + ax + b$, $a, b \in \mathbb{F}_p$, $4a^3 + 27b^2 \neq 0$.

Two Weierstrass equations might represent isomorphic curves.

Isomorphism classes

Two elliptic curves $E_{a,b}$ and $E_{a',b'}$ are isomorphic (over $\mathbb{F}_p$) **if and only if** $a' = \lambda^{-4}a$, $b' = \lambda^{-6}b$ for some $\lambda \in \mathbb{F}_p^\times$. The isomorphism between $E_{a,b}$ and $E_{a',b'}$ is given by

$$(x, y) \mapsto (\lambda^2 x, \lambda^3 y).$$

Each isomorphism class thus contains precisely $p - 1$ short Weierstrass equations.

The main result

All bits of the pairing-based OWF are hard-to-compute

If there is an oracle that predicts the kth bit of the input to f_Q on a significant fraction of all short Weierstrass equations in an isomorphism class then there is an efficient algorithm to invert f_Q.

The main result

All bits of the pairing-based OWF are hard-to-compute

If there is an oracle that predicts the kth bit of the input to f_Q on a significant fraction of all short Weierstrass equations in an isomorphism class then there is an efficient algorithm to invert f_Q.

Conclusion

Thus, if FAPI-2 is hard, all the bits of the input of the pairing-based OWF are hard-to-compute.

The result is in fact much more general as few properties of the pairing-based function f_Q are used.

The result is in fact much more general as few properties of the pairing-based function f_Q are used.

Bit Security for EC-based OWFs

Let $\mathbb{G}$ be an elliptic curve group and $f\colon \mathbb{G} \to \mathbb{G}_T$ be any function with the property that its definition is independent of the choice of short Weierstrass equation in the isomorphism class (e.g., the pairing-based OWF). Assuming that inverting f is hard, every bit of the input to f is secure.

Elliptic Curve-Based OWFs

The result is in fact much more general as few properties of the pairing-based function f_Q are used.

Bit Security for EC-based OWFs

Let $\mathbb{G}$ be an elliptic curve group and $f : \mathbb{G} \to \mathbb{G}_T$ be any function with the property that its definition is independent of the choice of short Weierstrass equation in the isomorphism class (e.g., the pairing-based OWF). Assuming that inverting f is hard, every bit of the input to f is secure.

Open Question: Are there other cryptographically interesting EC-based OWFs besides the pairing-based functions for which this result could apply?

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,
- Inverting the function $\Leftrightarrow$ list-decoding,

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,
- Inverting the function $\Leftrightarrow$ list-decoding,
- List-decoding via Fourier transforms:

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,
- Inverting the function $\Leftrightarrow$ list-decoding,
- List-decoding via Fourier transforms:
  - Codewords viewed as functions on $\mathbb{F}_p^\times$,

Outline of the Method

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,
- Inverting the function $\Leftrightarrow$ list-decoding,
- List-decoding via Fourier transforms:
  - Codewords viewed as functions on $\mathbb{F}_p^\times$,
  - Heavy Fourier coefficients: computation of heavy Fourier coefficients (a version of the SFT algorithm by Akavia–Goldwasser–Safra)

- Define a code - elliptic curve multiplication code (ECMC),
- Codewords are in bijection with the inputs to the OWF,
- Use oracle to get a noisy codeword close to the hidden input,
- Inverting the function $\Leftrightarrow$ list-decoding,
- List-decoding via Fourier transforms:
  - Codewords viewed as functions on $\mathbb{F}_p^\times$,
  - Heavy Fourier coefficients: computation of heavy Fourier coefficients (a version of the SFT algorithm by Akavia–Goldwasser–Safra)
  - Recoverability: for a given frequency, find all inputs having large Fourier coefficient at this frequency (a technique of Morillo–Ràfols).

Using the prediction oracle - naïve idea!

Suppose that we are given

Using the prediction oracle - naïve idea!

Suppose that we are given

- a hidden point $R \in \mathbb{G}$,

Using the prediction oracle - naïve idea!

Suppose that we are given

- a hidden point $R \in \mathbb{G}$,
- a short Weierstrass equation $W\colon y^2 = x^3 + ax + b$,

Suppose that we are given

- a hidden point $R \in \mathbb{G}$,
- a short Weierstrass equation $W \colon y^2 = x^3 + ax + b$,
- a prediction oracle $\mathcal{B}$ that returns a prediction $\mathcal{B}(W, f_Q(R))$ for the kth bit of the x-coordinate of the input point R on W.

Using the prediction oracle - naïve idea!

Suppose that we are given

- a hidden point $R \in \mathbb{G}$,
- a short Weierstrass equation $W\colon y^2 = x^3 + ax + b$,
- a prediction oracle $\mathcal{B}$ that returns a prediction $\mathcal{B}(W, f_Q(R))$ for the kth bit of the x-coordinate of the input point R on W.

Define a noisy codeword $w\colon \mathbb{F}_p^\times \to \{\pm 1\}$ as follows

$$w(\lambda) := \mathcal{B}(W_\lambda, f_Q(R)),$$

where $W_\lambda\colon y^2 = x^3 + \lambda^{-4}ax + \lambda^{-6}b$.

To each point R and each short Weierstrass equation W, one can associate a function (codeword) $C_R^W : \mathbb{F}_p^\times \to \{\pm 1\}$

$$C_R^W(\lambda) = B_k((R_{W_\lambda})_x) = B_k(\lambda^2 \cdot (R_W)_x),$$

where B_k returns $(-1)^b$ where b is the kth bit.

To each point R and each short Weierstrass equation W, one can associate a function (codeword) $C_R^W : \mathbb{F}_p^\times \to \{\pm 1\}$

$$C_R^W(\lambda) = B_k((R_{W_\lambda})_x) = B_k(\lambda^2 \cdot (R_W)_x),$$

where B_k returns $(-1)^b$ where b is the kth bit.

Properties needed for list-decoding?

Codewords and Points

To each point R and each short Weierstrass equation W, one can associate a function (codeword) $C_R^W : \mathbb{F}_p^\times \to \{\pm 1\}$

$$C_R^W(\lambda) = B_k((R_{W_\lambda})_x) = B_k(\lambda^2 \cdot (R_W)_x),$$

where B_k returns $(-1)^b$ where b is the kth bit.

Properties needed for list-decoding?

- Accessibility,

To each point R and each short Weierstrass equation W, one can associate a function (codeword) $C_R^W : \mathbb{F}_p^\times \to \{\pm 1\}$

$$C_R^W(\lambda) = B_k((R_{W_\lambda})_x) = B_k(\lambda^2 \cdot (R_W)_x),$$

where B_k returns $(-1)^b$ where b is the kth bit.

Properties needed for list-decoding?

- Accessibility,
- Fourier concentration,

Codewords and Points

To each point R and each short Weierstrass equation W, one can associate a function (codeword) $C_R^W : \mathbb{F}_p^\times \to \{\pm 1\}$

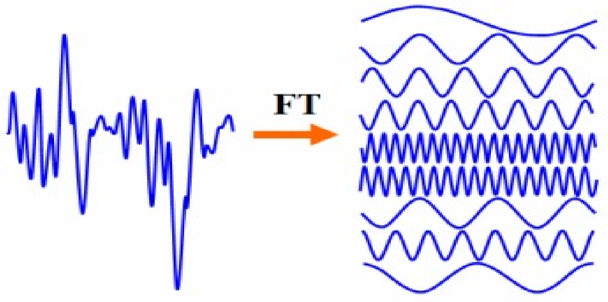
$$C_R^W(\lambda) = B_k((R_{W_\lambda})_x) = B_k(\lambda^2 \cdot (R_W)_x),$$

where B_k returns $(-1)^b$ where b is the kth bit.

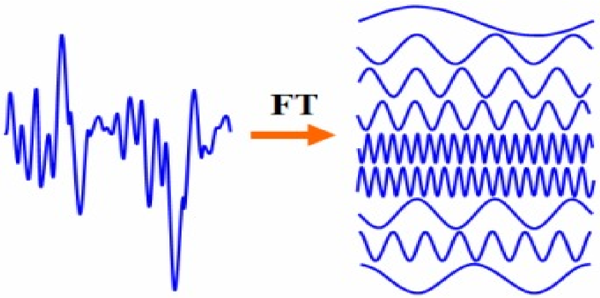
Properties needed for list-decoding?

- Accessibility,
- Fourier concentration,
- Recoverability.

Fourier Concentration

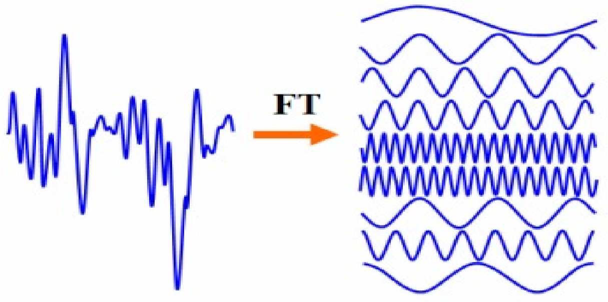


Fourier Concentration



- Fourier basis formed out of different frequencies (in our case, characters),

Fourier Concentration



- Fourier basis formed out of different frequencies (in our case, characters),
- A function is Fourier concentrated if the number of significant frequencies (characters) is small.

List Decoding

Recoverability

Given a frequency, find (in polynomial time) all codewords for which this frequency (character) is significant (i.e., has a large Fourier coefficient).

List Decoding

Recoverability

Given a frequency, find (in polynomial time) all codewords for which this frequency (character) is significant (i.e., has a large Fourier coefficient).

Fourier concentration + Recoverability $\Rightarrow$ List Decoding

Fourier Concentration and the First Attempt

Recall that

$$C_R^W(\lambda) = B_k(\lambda^2 \cdot (R_W)_x)$$

This will work fine if C_R^W were Fourier concentrated.

Fourier Concentration and the First Attempt

Recall that

$$C_R^W(\lambda) = B_k(\lambda^2 \cdot (R_W)_x)$$

This will work fine if C_R^W were Fourier concentrated.

- Estimating the Fourier coefficients of the codewords C_R^W reduces to estimating certain Gauss sums,

Fourier Concentration and the First Attempt

Recall that

$$C_R^W(\lambda) = B_k(\lambda^2 \cdot (R_W)_x)$$

This will work fine if C_R^W were Fourier concentrated.

- Estimating the Fourier coefficients of the codewords C_R^W reduces to estimating certain Gauss sums,
- Gauss sums estimates - classical in analytic number theory,

Fourier Concentration and the First Attempt

Recall that

$$C_R^W(\lambda) = B_k(\lambda^2 \cdot (R_W)_x)$$

This will work fine if C_R^W were Fourier concentrated.

- Estimating the Fourier coefficients of the codewords C_R^W reduces to estimating certain Gauss sums,
- Gauss sums estimates - classical in analytic number theory,
- Not clear how to show polynomially many significant Fourier coefficients, so following this natural approach is not feasible.

Fourier Concentration and the First Attempt

Recall that

$$C_R^W(\lambda) = B_k(\lambda^2 \cdot (R_W)_x)$$

This will work fine if C_R^W were Fourier concentrated.

- Estimating the Fourier coefficients of the codewords C_R^W reduces to estimating certain Gauss sums,
- Gauss sums estimates - classical in analytic number theory,
- Not clear how to show polynomially many significant Fourier coefficients, so following this natural approach is not feasible.

One needs a different list-decoding problem!

Using an idea of Boneh–Shparlinski:

The Elliptic Curve Multiplication Code (ECMC)

Using an idea of Boneh–Shparlinski:

- Define a new prediction oracle as follows:

$$\mathcal{B}'(W_\lambda, f_Q(R)) = \begin{cases} \mathcal{B}(W_{r(\lambda)}, f_Q(R)), & \text{if } \lambda \in \mathbb{F}_p^2 \\ \text{most probable value of } B_k(x) & \text{else,} \end{cases}$$

  where $r\colon \mathbb{F}_p^2 \to \mathbb{F}_p$ is a random square root function.

The Elliptic Curve Multiplication Code (ECMC)

Elliptic Curve Multiplication Code (ECMC)

Given a $W\colon y^2 = x^3 + ax + b$ representing E, define C_R^W as

$$C_R^W(\lambda) = B_k(\lambda \cdot (R_W)_x),$$

where R_W is the tuple (x, y) representing the point R on W.

Elliptic Curve Multiplication Code (ECMC)

Given a $W\colon y^2 = x^3 + ax + b$ representing E, define C_R^W as

$$C_R^W(\lambda) = B_k(\lambda \cdot (R_W)_x),$$

where R_W is the tuple (x, y) representing the point R on W.

- Fourier concentrated,

Elliptic Curve Multiplication Code (ECMC)

Given a $W\colon y^2 = x^3 + ax + b$ representing E, define C_R^W as

$$C_R^W(\lambda) = B_k(\lambda \cdot (R_W)_x),$$

where R_W is the tuple (x, y) representing the point R on W.

- Fourier concentrated,
- Recoverable (a technique of Morillo–Ràfols).

Summary of Results:

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,
- In particular, input bits to FA pairing-based OWFs are hard-to-compute,

Summary and Open Questions

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,
- In particular, input bits to FA pairing-based OWFs are hard-to-compute,

Open Questions:

Summary and Open Questions

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,
- In particular, input bits to FA pairing-based OWFs are hard-to-compute,

Open Questions:

- Same result, but on a fixed Weierstrass equation,

Summary and Open Questions

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,
- In particular, input bits to FA pairing-based OWFs are hard-to-compute,

Open Questions:

- Same result, but on a fixed Weierstrass equation,
- Assuming imperfect oracle on an isogeny class of curves (curves having a fixed number of points) - work in progress,

Summary of Results:

- Every bit to the input of any EC-based OWF is hard-to-compute,
- In particular, input bits to FA pairing-based OWFs are hard-to-compute,

Open Questions:

- Same result, but on a fixed Weierstrass equation,
- Assuming imperfect oracle on an isogeny class of curves (curves having a fixed number of points) - work in progress,
- Assuming prediction of blocks of bits.