Hashing with Elliptic Curve *L*-Functions

Sami Omar[1,2], Raouf Ouni[1] and Saber Bouanani [1]

[1] Faculty of Science of Tunis, Tunisia
[2] King Khalid University, Abha, Saudi Arabia

International Workshop on the Arithmetic of Finite Fields
WAIFI 2012

Plan

Introduction

Secure communication needs :

- Authenticity
- Integrity of information
- Non Repudiation

For this purpose :

- Cryptographic tools are precious :
  - Hash functions
  - Message Authentication Code (MAC)

Introduction

Secure communication needs :

- Authenticity
- Integrity of information
- Non Repudiation

For this purpose :

- Cryptographic tools are precious :
  - Hash functions
  - Message Authentication Code (MAC)

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.
- The output has a fixed length.

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.
- The output has a fixed length.
- $H(x)$ is relatively easy to compute for any given x.

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.
- The output has a fixed length.
- $H(x)$ is relatively easy to compute for any given x.
- $H(x)$ is one-way.

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.
- The output has a fixed length.
- $H(x)$ is relatively easy to compute for any given x.
- $H(x)$ is one-way.
- $H(x)$ is collision-free.

Hash functions

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value $h = H(m)$.
- The input can be of any length.
- The output has a fixed length.
- $H(x)$ is relatively easy to compute for any given x.
- $H(x)$ is one-way.
- $H(x)$ is collision-free.

Hash functions

MAC

A Message Authentication Code or MAC is a function h that satisfies the following conditions:

1. The input X can be of arbitrary length and the result $h(K;X)$ has a fixed length of n bits. The function has as secondary input the key K, with a fixed length of k bits.

Hash functions

MAC

A Message Authentication Code or MAC is a function h that satisfies the following conditions:

1. The input X can be of arbitrary length and the result $h(K;X)$ has a fixed length of n bits. The function has as secondary input the key K, with a fixed length of k bits.
2. Given h, K and an input X, the computation of $h(K;X)$ must be 'easy'.

Hash functions

MAC

A Message Authentication Code or MAC is a function h that satisfies the following conditions:

1. The input X can be of arbitrary length and the result $h(K;X)$ has a fixed length of n bits. The function has as secondary input the key K, with a fixed length of k bits.
2. Given h, K and an input X, the computation of $h(K;X)$ must be 'easy'.
3. Given a message X (but with unknown K), it must be 'hard' to determine $h(K;X)$. Even when a large set of pairs $X_i; h(K;X_i)$ is known, it is 'hard' to determine the key K or to compute $h(K;X')$ for any new message $X' \neq X_i, \forall i$.

Hash functions

MAC

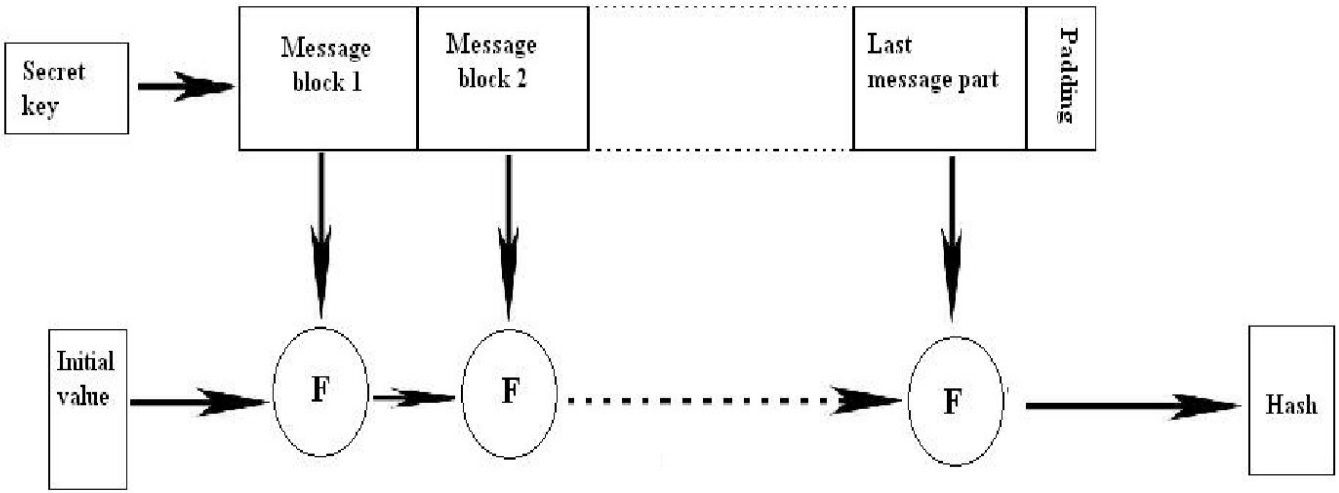
A Message Authentication Code or MAC is a function h that satisfies the following conditions:

1. The input X can be of arbitrary length and the result $h(K;X)$ has a fixed length of n bits. The function has as secondary input the key K, with a fixed length of k bits.
2. Given h, K and an input X, the computation of $h(K;X)$ must be 'easy'.
3. Given a message X (but with unknown K), it must be 'hard' to determine $h(K;X)$. Even when a large set of pairs $X_i; h(K;X_i)$ is known, it is 'hard' to determine the key K or to compute $h(K;X')$ for any new message $X' \neq X_i, \forall i$.

Construction of hash function

Construction of hash function

- Merkle-Damgard propose an iterative way to construct hash function from collision-resistant one-way compression functions.
- This construction was used in the design of many popular hash algorithms such as MD5, SHA1 and SHA2.



Elliptic curve

- An elliptic curve over $\mathbb{K}$ is defined by the equation

$$E : y^2 = x^3 + ax + b, \qquad a, b \in \mathbb{K},$$

- The discriminant $\Delta(E)$ of E is defined by $\Delta(E) = -4a^3 - 27b^2$.

Mordell-Weil theorem

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} | y^2 = x^3 + ax + b\} \cup O$$

is a finite abelian group with O is a point at ∞ as the identity element.

Elliptic curve

- An elliptic curve over $\mathbb{K}$ is defined by the equation

$$E : y^2 = x^3 + ax + b, \qquad a, b \in \mathbb{K},$$

- The discriminant $\Delta(E)$ of E is defined by $\Delta(E) = -4a^3 - 27b^2$.

Mordell-Weil theorem

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} | y^2 = x^3 + ax + b\} \cup O$$

is a finite abelian group with O is a point at ∞ as the identity element.

Elliptic Curve *L*-Function

- In elliptic curve cryptography, we are interested in case where $\mathbb{K}$ is a finite field, $\mathbb{K} = \mathbb{F}_q$, q is a prime number power.
- For each prime p, we define

$$a_p = p + 1 - \sharp E(\mathbb{F}_p)$$

where $\mathbb{F}_p$ is a finite field with p elements and $\sharp E(\mathbb{F}_p)$ is the order of $E(\mathbb{F}_p) = O \cup \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p | y^2 = x^3 + ax + b\}$

Elliptic Curve *L*-Function

- In elliptic curve cryptography, we are interested in case where $\mathbb{K}$ is a finite field, $\mathbb{K} = \mathbb{F}_q$, q is a prime number power.
- For each prime p, we define

$$a_p = p + 1 - \sharp E(\mathbb{F}_p)$$

where $\mathbb{F}_p$ is a finite field with p elements and $\sharp E(\mathbb{F}_p)$ is the order of $E(\mathbb{F}_p) = O \cup \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p | y^2 = x^3 + ax + b\}$

- The Hasse theorem says that for each prime p, a_p satisfies the following inequality

$$|a_p| \leq 2\sqrt{p}$$

Elliptic Curve *L*-Function

- In elliptic curve cryptography, we are interested in case where $\mathbb{K}$ is a finite field, $\mathbb{K} = \mathbb{F}_q$, q is a prime number power.
- For each prime p, we define

$$a_p = p + 1 - \sharp E(\mathbb{F}_p)$$

where $\mathbb{F}_p$ is a finite field with p elements and $\sharp E(\mathbb{F}_p)$ is the order of $E(\mathbb{F}_p) = O \cup \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p | y^2 = x^3 + ax + b\}$

- The Hasse theorem says that for each prime p, a_p satisfies the following inequality

$$|a_p| \leq 2\sqrt{p}$$

Elliptic Curve *L*-Function

- When p divides $\Delta(E)$, one finds $a_p = 0, \pm 1$ where the exact value is determined according to the type of singularity of E modulo p.
- The L-function attached to E is defined as

$$L(s,E) = \prod_{p|\Delta(E)} \frac{1}{1 - a_p p^{-s}} \prod_{p\nmid\Delta(E)} \frac{1}{1 - a_p p^{-s} + p^{1-2s}}.$$

Elliptic Curve L-Function

- When p divides $\Delta(E)$, one finds $a_p = 0, \pm 1$ where the exact value is determined according to the type of singularity of E modulo p.
- The L-function attached to E is defined as

$$L(s,E) = \prod_{p|\Delta(E)} \frac{1}{1 - a_p p^{-s}} \prod_{p\nmid\Delta(E)} \frac{1}{1 - a_p p^{-s} + p^{1-2s}}.$$

- By the Hasse theorem, one can show that the Euler product $L(s,E)$ converges for all s with $Re(s) > \frac{3}{2}$ and it is given by the following absolutely convergent series

$$L(s,E) = \sum_{n=1}^{\infty} a_n n^{-s}.$$

Elliptic Curve L-Function

- When p divides $\Delta(E)$, one finds $a_p = 0, \pm 1$ where the exact value is determined according to the type of singularity of E modulo p.
- The L-function attached to E is defined as

$$L(s,E) = \prod_{p|\Delta(E)} \frac{1}{1 - a_p p^{-s}} \prod_{p\nmid\Delta(E)} \frac{1}{1 - a_p p^{-s} + p^{1-2s}}.$$

- By the Hasse theorem, one can show that the Euler product $L(s,E)$ converges for all s with $Re(s) > \frac{3}{2}$ and it is given by the following absolutely convergent series

$$L(s,E) = \sum_{n=1}^{\infty} a_n n^{-s}.$$

ECOH

- ECOH divides the message into blocks, maps each block to an elliptic curve point and adds these points together with two more points. One additional point depends on the message length and the exclusive-or of all message blocks.
- given n message blocks $M_0, M_1, \ldots, M_{n-1}$

$$P_i := P(M_i, i) \quad \text{for } i = 0, \ldots, n-1$$

P is a function that maps a message block and an integer to an elliptic curve point.

ECOH

- ECOH divides the message into blocks, maps each block to an elliptic curve point and adds these points together with two more points. One additional point depends on the message length and the exclusive-or of all message blocks.
- given n message blocks $M_0, M_1, \ldots, M_{n-1}$

$$P_i := P(M_i, i) \quad \textit{for } i = 0, \ldots, n-1$$

P is a function that maps a message block and an integer to an elliptic curve point.

ECOH

- $$X_1 := P'(n), \quad X_2 := P''(\bigoplus_{i=0}^{n-1} M_i, n)$$
  - P' computes the padding point which depends only on the length of M.
  - P'' computes the checksum point X_2 which depends on the exclusive-or of all the message blocks and on the length of M.
- $$Q := \sum_{i=0}^{n-1} P_i + X_1 + X_2, \quad R := f(Q)$$

The result Q is passed through an output transformation function f to get the hash result R.

ECOH

- $$X_1 := P'(n), \quad X_2 := P''\left(\bigoplus_{i=0}^{n-1} M_i, n\right)$$
  - P' computes the padding point which depends only on the length of M.
  - P'' computes the checksum point X_2 which depends on the exclusive-or of all the message blocks and on the length of M.
- $$Q := \sum_{i=0}^{n-1} P_i + X_1 + X_2, \quad R := f(Q)$$

  The result Q is passed through an output transformation function f to get the hash result R.

One way function

- For a given elliptic curve E over $\mathbb{Q}$, the coefficients a_p of $L(s, E)$ can be computed in polynomial time.
- Nevertheless, it is not easy to go in the other direction.

One way function

- For a given elliptic curve E over $\mathbb{Q}$, the coefficients a_p of $L(s, E)$ can be computed in polynomial time.
- Nevertheless, it is not easy to go in the other direction.
- let T be a very large number and let α, β be fixed positive constants.

One way function

- For a given elliptic curve E over $\mathbb{Q}$, the coefficients a_p of $L(s, E)$ can be computed in polynomial time.
- Nevertheless, it is not easy to go in the other direction.
- let T be a very large number and let α, β be fixed positive constants.
- Let we denote by $\mathcal{L}^T$ the set of all elliptic curves E over $\mathbb{Q}$ with $T \leq \Delta(E) \leq 2T$ and let k and m be integers satisfying

$$(\log T)^{\alpha} \leq k, \quad m \leq (\log T)^{\beta}.$$

One way function

- For a given elliptic curve E over $\mathbb{Q}$, the coefficients a_p of $L(s, E)$ can be computed in polynomial time.
- Nevertheless, it is not easy to go in the other direction.
- let T be a very large number and let α, β be fixed positive constants.
- Let we denote by $\mathcal{L}^T$ the set of all elliptic curves E over $\mathbb{Q}$ with $T \leq \Delta(E) \leq 2T$ and let k and m be integers satisfying

$$(\log T)^{\alpha} \leq k, \quad m \leq (\log T)^{\beta}.$$

One way function

Conjecture

The following map

$$H: \quad \mathcal{L}^T \to \mathbb{Q}^{k+1}$$

$$E \mapsto (a_m, a_{m+1}, \ldots, a_{m+k})$$

where $L(s,E) = \sum_{n=1}^{\infty} a_n n^{-s}$, is a one way function.

One way function : Application

Authentication

1. Both Alice and Bob are in possesion of a secret key E which is an elliptic curve.
2. Bob sends to Alice randomly chosen integers m and $b > 0$.

One way function : Application

Authentication

1. Both Alice and Bob are in possesion of a secret key E which is an elliptic curve.
2. Bob sends to Alice randomly chosen integers m and $b > 0$.
3. Alice computes and sends to Bob the vector $v = (a_m, a_{m+1}, \ldots, a_{m+b})$.

One way function : Application

Authentication

1. Both Alice and Bob are in possesion of a secret key E which is an elliptic curve.
2. Bob sends to Alice randomly chosen integers m and $b > 0$.
3. Alice computes and sends to Bob the vector $v = (a_m, a_{m+1}, \ldots, a_{m+b})$.
4. Bob checks if Alice's list is correct then she is an authenticated user.

One way function : Application

Authentication

1. Both Alice and Bob are in possesion of a secret key E which is an elliptic curve.
2. Bob sends to Alice randomly chosen integers m and $b > 0$.
3. Alice computes and sends to Bob the vector $v = (a_m, a_{m+1}, \ldots, a_{m+b})$.
4. Bob checks if Alice's list is correct then she is an authenticated user.

MAC's elliptic curve L-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.

MAC's elliptic curve L-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.
3. $H_0 = IV = (a_k, a_{k+1}, \ldots, a_{k+r})$, where $(a_k, a_{k+1}, \ldots, a_{k+r})$ are r-coefficients in stage i of the L-function of the secret elliptic curve.

MAC's elliptic curve L-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.
3. $H_0 = IV = (a_k, a_{k+1}, \ldots, a_{k+r})$, where $(a_k, a_{k+1}, \ldots, a_{k+r})$ are r-coefficients in stage i of the L-function of the secret elliptic curve.
4. ***For each*** $i \in \ 1 \leq i \leq t+1$
$$H_i = f(H_{i-1}, x_i) = x_i \oplus \pi(a_{k_i}, a_{k_i+1}, \ldots, a_{k_i+r}).$$

MAC's elliptic curve L-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.
3. $H_0 = IV = (a_k, a_{k+1}, \ldots, a_{k+r})$, where $(a_k, a_{k+1}, \ldots, a_{k+r})$ are r-coefficients in stage i of the L-function of the secret elliptic curve.
4. ***For each*** $i \in \ 1 \leq i \leq t+1$
   $H_i = f(H_{i-1}, x_i) = x_i \oplus \pi(a_{k_i}, a_{k_i+1}, \ldots, a_{k_i+r})$.
5. $h(x) = H_{t+1}$.

MAC's elliptic curve L-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.
3. $H_0 = IV = (a_k, a_{k+1}, \ldots, a_{k+r})$, where $(a_k, a_{k+1}, \ldots, a_{k+r})$ are r-coefficients in stage i of the L-function of the secret elliptic curve.
4. ***For each*** $i \in \ 1 \leq i \leq t+1$
   $H_i = f(H_{i-1}, x_i) = x_i \oplus \pi(a_{k_i}, a_{k_i+1}, \ldots, a_{k_i+r})$.
5. $h(x) = H_{t+1}$.
6. RETURN a hash value of the message x

MAC's elliptic curve *L*-function protocol

1. Given a secret key (E, k, π) and a message $x = x_1 x_2 \ldots x_{t+1}$ where we divide the message x into $t+1$ blocks with size length r, k is a positive number and π is a permutation.
2. Use a padding algorithm with input x.
3. $H_0 = IV = (a_k, a_{k+1}, \ldots, a_{k+r})$, where $(a_k, a_{k+1}, \ldots, a_{k+r})$ are r-coefficients in stage i of the L-function of the secret elliptic curve.
4. ***For each*** $i \in \ 1 \leq i \leq t+1$
   $H_i = f(H_{i-1}, x_i) = x_i \oplus \pi(a_{k_i}, a_{k_i+1}, \ldots, a_{k_i+r})$.
5. $h(x) = H_{t+1}$.
6. RETURN a hash value of the message x

Security

- Let the projection map

$$H_1 : \quad \mathcal{L}_C^B \to \mathbb{Q}^{k+1}$$

$$E \mapsto (a_m, a_{m+1}, \ldots, a_{m+k})$$

where $L(s, E) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}$.

- And let C a fixed positive constant with $C \geq \frac{\log T}{T}$.

Security

- Let the projection map

$$H_1 : \quad \mathcal{L}_C^B \to \mathbb{Q}^{k+1}$$

$$E \mapsto (a_m, a_{m+1}, \ldots, a_{m+k})$$

  where $L(s, E) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}$.
- And let C a fixed positive constant with $C \geq \frac{\log T}{T}$.
- Let denote $\mathcal{L}_C^T$ the subset of $\mathcal{L}^T$ consisting of all elliptic curves E: $y^2 = x^3 + ax + b$, $a, b \in \mathbb{Q}$ and $0 \leq |a|, |b| \leq CT$.

Security

- Let the projection map

$$H_1 : \quad \mathcal{L}_C^B \to \mathbb{Q}^{k+1}$$

$$E \mapsto (a_m, a_{m+1}, \ldots, a_{m+k})$$

  where $L(s, E) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}$.
- And let C a fixed positive constant with $C \geq \frac{\log T}{T}$.
- Let denote $\mathcal{L}_C^T$ the subset of $\mathcal{L}^T$ consisting of all elliptic curves E: $y^2 = x^3 + ax + b$, $a, b \in \mathbb{Q}$ and $0 \leq |a|, |b| \leq CT$.

Theorem

For any $(c_m, c_{m+1}, \ldots, c_{m+k}) \in \mathbb{Q}^{k+1}$ *satisfying* $|c_n| \leq \sqrt{n}\, d(n)$, *here* $d(n)$ *is the number of positive divisors of n. And for any* $\varepsilon > 0$, *there exits an algorithm which computes an elliptic curve E in* $\mathcal{L}_C^T$ *such that* $a_p = c_p$ *for all primes* $m \leq p \leq m + k$ *with a running time of order* $(CT)^{\frac{3}{2}+\varepsilon}$.

Conclusion

- In this work, we proposed a simple and efficient MAC function based on the *L*-function attached to a given elliptic curve.
- It will be important to carry out the implementation aspect and complexity of this hashing protocol on hardware and software platforms.

Conclusion

- In this work, we proposed a simple and efficient MAC function based on the L-function attached to a given elliptic curve.
- It will be important to carry out the implementation aspect and complexity of this hashing protocol on hardware and software platforms.
- It may be also useful to do the same construction in the case of L-functions attached to hyperelliptic curves.

Conclusion

- In this work, we proposed a simple and efficient MAC function based on the *L*-function attached to a given elliptic curve.
- It will be important to carry out the implementation aspect and complexity of this hashing protocol on hardware and software platforms.
- It may be also useful to do the same construction in the case of *L*-functions attached to hyperelliptic curves.

THANK YOU