SM9 identity-based cryptographic algorithms
Part 1: General

Contents

SM9 identity-based cryptographic algorithms

Part 1: General

1 Scope

This part describes fundamental mathematical knowledge and cryptographic techniques necessary for implementing cryptographic mechanisms provided in other parts of this standard.

This standard is applicable to the implementation, application and testing of commercial identity-based cryptographic algorithms.

This standard applies to the elliptic curves over the finite field F_p, where p is a prime number that satisfies $p > 2^{191}$.

2 Terms and definitions

2.1 identity

information that can be used to confirm the identity of an entity, composed of non-repudiable information about the entity, such as its distinguished name, email address, identity card number, and telephone number

2.2 master key

topmost key in the key hierarchy of an identity–based cryptographic system, composed of the master private key and master public key. The master public key is publicly available, while the master private key is preserved by the KGC in secrecy. A user's private key is generated by the KGC using the master private key and the user's identity. In an identity–based cryptographic system, the master private key is usually generated by the KGC using random number generators; the master public key is generated with the master private key and system parameters.

This standard specifies a different master key for the signature system than that of the encryption system. The master key of the digital signature algorithm, which belongs to the signature system, is the signature master key. The master key of the key exchange protocol, key encapsulation mechanism and public key encryption algorithm, which all belong to the encryption system, is the encryption master key.

2.3 key generation center (KGC)

trusted authority responsible for the selection of the system parameters, generation of master keys and generation of users' private keys within SM9 identity-based cryptographic algorithms

3 Symbols and abbreviations

The following symbols and abbreviations apply to this part.

cf: cofactor of the order of an elliptic curve relative to N

cid: curve identifier used to distinguish the type of elliptic curve used, denoted by one byte

DLP: discrete logarithm problem over finite fields

$\deg(f)$: the degree of the polynomial $f(x)$

d_1, d_2: two divisors of k

E: an elliptic curve over finite fields

ECDLP: discrete logarithm problem over elliptic curves

$E(F_q)$: a set consisting of all rational points (including the point at infinity O) of the elliptic curve E over the finite field F_q

$E(F_q)[r]$: the set of r-torsion points in $E(F_q)$, that is the torsion subgroup of $E(F_q)$ of order r

e: a bilinear pairing from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$

eid: bilinear pairing identifier used to distinguish the type of bilinear pairing used, denoted by one byte

F_p: a prime field with p elements

F_q: a finite field with q elements

F_q^*: the multiplicative group composed of all the nonzero elements in F_q

F_{q^m}: the m-dimensional extension field of the finite field F_q

$\mathbb{G}_T$: a multiplicative cyclic group of prime order N

$\mathbb{G}_1$: an additive cyclic group of prime order N

$\mathbb{G}_2$: an additive cyclic group of prime order N

$\gcd(x, y)$: the greatest common divisor of x and y

k: the embedding degree of the curve $E(F_q)$ relative to N, where N is a prime factor of $\#E(F_q)$

m: the degree of the finite field extension F_{q^m}/F_q

mod $f(x)$: the operation of modulo the polynomial $f(x)$

mod n: the operation of modulo n, for example, $23 \bmod 7 = 2$

N: the order of the cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, which is a prime number greater than 2^{191}

O: the point at infinity or the zero point on an elliptic curve, which is the identity element of the elliptic curve additive group

P: $P = (x_P, y_P)$ is a nonzero point on an elliptic curve, where its coordinates x_P and y_P satisfy the elliptic curve equation

P_1: a generator of $\mathbb{G}_1$

P_2: a generator of $\mathbb{G}_2$

$P + Q$: addition of two points P and Q on the elliptic curve E

p: a prime number greater than 2^{191}

q: the number of elements in the finite field F_q

x_P: the x-coordinate of point P

$x||y$: the concatenation of x and y, where x and y are bit strings or byte strings

$x \equiv y \pmod q$: x and y are congruent modulo q, that is $x \bmod q = y \bmod q$

y_P: the y-coordinate of point P

$\#E(K)$: the number of points in $E(K)$, also called the order of the elliptic curve group $E(K)$, where K is a finite field (including F_q and F_{q^k})

$\langle P \rangle$: the cyclic group generated by the point P on an elliptic curve

$[u]P$: the u multiple of a point P on an elliptic curve

$[x, y]$: the set of integers which are not less than x and not greater than y

$\lceil x \rceil$: ceiling function that maps to the smallest integer not less than x, for example, $\lceil 7 \rceil = 7$, $\lceil 8.3 \rceil = 9$

$\lfloor x \rfloor$: floor function that maps to the largest integer not greater than x, for example, $\lfloor 7 \rfloor = 7$, $\lfloor 8.3 \rfloor = 8$

β: twisted curve parameter

Ψ: a homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ satisfying $P_1 = \Psi(P_2)$

$\oplus$: the bitwise XOR operator that operates on two bit strings of the same length

4 Finite field and elliptic curve

4.1 Finite field

4.1.1 Overview

A field consists of a non-empty set F with two operations: the addition (denoted by "+") and the multiplication (denoted by "·").

It satisfies following properties:
a) $(F, +)$ is an additive abelian group, in which 0 denotes the identity element.
b) $(F \backslash \{0\}, \cdot)$ is a multiplicative abelian group, in which 1 denotes the identity element.
c) Distributive law: $(a + b)c = ac + bc$ for all $a, b, c \in F$.

If F is a finite set, then the field is called a finite field. The number of elements in the finite field is called the order of the finite field.

4.1.2 Prime field F_p

When the order of a finite field is prime, we call the field a prime field.

Let p be a prime number, then the residue of integers modulo p, $\{0,1, \ldots p-1\}$, with respect to the addition modulo p and the multiplication modulo p can construct a prime field of order p, denoted by F_p.

F_p has the following properties:

a) the additive identity element is 0.

b) the multiplicative identity element is 1.

c) the addition of field elements is that of integers modulo p, namely, if $a, b \in F_p$, then $a + b = (a + b) \bmod p$.

d) the multiplication of field elements is that of integers modulo p, namely, if $a, b \in F_p$, then $a \cdot b = (a \cdot b) \bmod p$.

4.1.3 Finite field F_{q^m}

Let q be a prime or a prime power, $f(x)$ be an m-degree $(m > 1)$ irreducible polynomial (reduced polynomial or field polynomial) in the polynomial ring $F_q[x]$, quotient ring $F_q[x]/(f(x))$ be a finite field with q^m elements (denoted by F_{q^m}), then F_{q^m} is the extension field of F_q, F_q is the subfield of F_{q^m}, and m is the extension degree. F_{q^m} can be seen as the m-dimensional vector space of F_q and its elements can be uniquely represented by $a_0\beta_0 + a_1\beta_1 + \cdots a_{m-1}\beta_{m-1}$, where $a_i \in F_q$, $\beta_0, \dots, \beta_{m-1}$ is a base of F_{q^m} over F_q.

The elements of F_{q^m} can be represented via polynomial basis or normal basis. In this standard, unless otherwise specified, all elements of F_{q^m} are represented by polynomial basis.

Choose a monic irreducible polynomial $f(x) = x^m + f_{m-1}x_{m-1} + \cdots + f_2x^2 + f_1x + f_0$ $(f_i \in F_q,\ i = 0,1, \dots, m-1)$, then F_{q^m} is composed of all polynomials in the polynomial ring $F_q[x]$ of degree less than m. The set of polynomials $\{x^{m-1}, x^{m-2}, \dots, x, 1\}$ is a base for F_{q^m} over F_q, which is called a polynomial basis. For any element $a(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ in F_{q^m}, its coefficients over F_q constitute an m-dimensional vector, denoted by $a = (a_{m-1}, a_{m-2}, \dots, a_1, a_0)$, where $a_i \in F_q, i = 0,1, \dots, m-1$.

F_{q^m} has the following properties:

a) The zero element 0 is represented by an m-dimensional vector (0,0, ... 0,0,0).

b) The multiplicative identity element is represented by an m-dimensional vector (0,0, ... 0,0,1).

c) The addition of two field elements is the addition of vectors, and each vector component adopts addition of field F_q.

d) The multiplication of elements a and b is defined like this: let a and b correspond to the polynomials $a(x)$ and $b(x)$ over F_q respectively; then, $a \cdot b$ is defined as the corresponding vector of the polynomial $(a(x) \cdot b(x)) \bmod f(x)$.

e) The inverse element: suppose $a(x)$ is the corresponding polynomial of a over F_q, $a^{-1}(x)$ is the corresponding polynomial of a^{-1} over F_q, such that $a(x) \cdot a^{-1}(x) = 1 \bmod f(x)$.

See Annex A.1 for more details about F_{q^m}.

4.2 Elliptic curves over finite field

The elliptic curve over finite field F_{q^m} $(m \geq 1)$ is a set of points. A point P (except the point O) on the elliptic curve can be represented by the coordinates $P = (x_P, y_P)$, where x_P and y_P are field elements satisfying a certain equation, and are called the x-coordinate and y-coordinate, respectively.

This part describes elliptic curves whose characteristic is a large prime p.

In this part, the points on an elliptic curve are represented by affine coordinates, unless otherwise specified.

The equation of elliptic curves defined over F_{q^m} is:

$$y^2 = x^3 + ax + b, a, b \in F_{p^m}, \text{and } 4a^3 + 27b^2 \neq 0. \tag{1}$$

The elliptic curve $E(F_{q^m})$ is defined as:

$$E(F_{q^m}) = \{(x, y) \mid x, y \in F_{q^m}, \text{satisfying the equation (1)}\} \cup \{O\}, \text{where } O \text{ is the point at infinity.}$$

The number of points on the elliptic curve $E(F_{q^m})$ is represented by $\#E(F_{q^m})$, which is also called the order of $E(F_{q^m})$.

This standard requires the prime $p > 2^{191}$.

Let E and E' be elliptic curves defined over F_q. If there exists an isomorphic map $\phi_d: E'(F_{q^d}) \to E(\mathrm{F}_{q^{\mathrm{d}}})$, where d is the smallest integer which makes the map exist, then E' is called the degree d twisted curve of E. There are three cases of the value of d when $p \geq 5$:

a) If $a = 0, b \neq 0$, then $d = 6$, and $E': \mathrm{y}^2 = x^3 + \beta b$, $\phi_6: E' \to E: (x, y) \mapsto (\beta^{-1/3}x, \beta^{-1/2}y)$.
b) If $b = 0, a \neq 0$, then $d = 4$, and $E': \mathrm{y}^2 = x^3 + \beta ax$, $\phi_4: E' \to E: (x, y) \mapsto (\beta^{-1/2}x, \beta^{-3/4}y)$.
c) If $a \neq 0, b \neq 0$, then $d = 2$, and $E': \mathrm{y}^2 = x^3 + \beta^2 ax + \beta^3 b$, $\phi_2: E' \to E: (x, y) \mapsto (\beta^{-1}x, \beta^{-3/2}y)$.

4.3 Elliptic curve group

The points on elliptic curve $E(F_{q^m})$, where $(m \geq 1)$, constitute an abelian group based on the following addition operation rules:

a) $O + O = O$.
b) $\forall P = (x, y) \in E(F_{q^m}) \backslash \{O\}, P + O = O + P = P$.
c) $\forall P = (x, y) \in E(F_{q^m}) \backslash \{O\}$, the inverse element of P is $-P = (x, -y)$, and $P + (-P) = O$.
d) The addition rules for two different points (wherein these points are not the inverse of each other):

Let $P_1 = (x_1, y_1) \in E(F_{q^m}) \backslash \{O\}$, $P_2 = (x_2, y_2) \in E(F_{q^m}) \backslash \{O\}$, and $x_1 \neq x_2$.
Let $P_3 = (x_3, y_3) = P_1 + P_2$, then

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases}$$

where

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

e) Point doubling:

Let $P_1 = (x_1, y_1) \in E(F_{q^m}) \backslash \{O\}$, and $y_1 \neq 0$, $P_3 = (x_3, y_3) = P_1 + P_1$, then

$$\begin{cases} x_3 = \lambda^2 - 2x_1, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases}$$

where

$$\lambda = \frac{3x_1^2 + a}{2y_1}.$$

4.4 Scalar multiplication on elliptic curve

The repeated addition of the same point is called the scalar multiplication of the point. Let u be a positive integer, P be a point on the elliptic curve, then the u multiple of the point P is denoted by $Q = [u]P = \underbrace{P + P + \cdots + P}_{u\ P's}$.

Scalar multiplication can be extended to 0-multiple and negative-multiple operations: $[0]P = O$, $[-u]P = [u](-P)$.

Scalar multiplication can be calculated efficiently using certain techniques; please refer to Annex A.2 for them.

4.5 Verification of points in a subgroup of an elliptic curve

Input: The parameters a and b which define the elliptic curve equation over F_{q^m}, where q is an odd prime and $m \geq 1$, the order N of the subgroup $\mathbb{G}$ of the elliptic curve $E(F_{q^m})$, a pair of elements in F_{q^m} (x, y).

Output: If (x, y) is an element of the group $\mathbb{G}$, then output "valid", otherwise output "invalid".

a) Check if (x, y) satisfies the equation of the elliptic curve $y^2 = x^3 + ax + b$.
b) Let $Q = (x, y)$, check if $[N]Q = O$.

If any of these above verification fail, output "invalid", otherwise output "valid".

4.6 Discrete logarithm problem

4.6.1 Discrete logarithm problem over finite field

The set of all nonzero elements in F_{q^m} (q is an odd prime, $m \geq 1$) forms a multiplicative cyclic group, denoted by $F_{q^m}^*$. An element $g \in F_{q^m}^*$ is called a generator if it satisfies $F_{q^m}^* = \{g^i \mid 0 \leq i \leq q^m - 2\}$. The minimal integer t such that $a^t = 1$ is called the order of a in $F_{q^m}^*$. The order of $F_{q^m}^*$ is $q^m - 1$, so $t \mid q^m - 1$.

Suppose the generator of $F_{q^m}^*$ is g, $y \in F_{q^m}^*$, the discrete logarithm problem over a finite field is to find the integer $x \in [0, q^m - 1]$ such that $y = g^x$ in $F_{q^m}^*$.

4.6.2 Elliptic curve discrete logarithm problem (ECDLP)

For an elliptic curve $E(F_{q^m})$ $(m \geq 1)$, the point $P \in E(F_{q^m})$ of order n and $Q \in \langle P \rangle$, ECDLP is to find $l \in [0, n-1]$ satisfying $Q = [l]P$.

5 Bilinear pairings and secure curves

5.1 Bilinear pairings

Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \cdot)$ be three cyclic groups. The order of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ is a prime N, P_1 is a generator of $\mathbb{G}_1$, P_2 is a generator of $\mathbb{G}_2$, and there exists a homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ such that $\psi(P_2) = P_1$.

Bilinear pairing e is a map of $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying the following conditions:
a) Bilinearity: for any $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $a, b \in \mathbb{Z}_N$, $e([a]P, [b]Q) = e(P, Q)^{ab}$.

b) Non-degeneracy: $e(P_1, P_2) \neq 1_{\mathbb{G}_T}$.
c) Computability: for any $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, there exists an efficient algorithm to compute $e(P, Q)$.

Bilinear pairings used in this part are defined on elliptic curve groups, such as the Weil pairing, the Tate pairing, the Ate pairing and the R-ate pairing.

5.2 Security

The security of bilinear pairings is based on the following hard problems:

Problem 1 (Bilinear Inverse Diffie-Hellman Problem, BIDH) For $a, b \in [1, N-1]$, given $[a]P_1$, $[b]P_2$, it is hard to compute $e(P_1, P_2)^{b/a}$.

Problem 2 (Decisional Bilinear Inverse Diffie-Hellman Problem, DBIDH) For $a, b, r \in [1, N-1]$, it is hard to distinguish $(P_1, P_2, [a]P_1, [b]P_2, e(P_1, P_2)^{b/a})$ from $(P_1, P_2, [a]P_1, [b]P_2, e(P_1, P_2)^r)$.

Problem 3 (τ-Bilinear Inverse Diffie-Hellman Problem, τ-BDHI) For integer τ and $x \in [1, N-1]$, given $(P_1, [x]P_1, P_2, [x]P_2, [x^2]P_2, \dots, [x^\tau]P_2)$, it is hard to compute $e(P_1, P_2)^{1/x}$.

Problem 4 (τ-Gap-Bilinear Inverse Diffie-Hellman Problem, τ-Gap-BDHI) For integer τ and $x \in [1, N-1]$, given $(P_1, [x]P_1, P_2, [x]P_2, [x^2]P_2, \dots, [x^\tau]P_2)$ and the DBIDH algorithm, it is hard to compute $e(P_1, P_2)^{1/x}$.

The security of the SM9 identity-based cryptographic algorithms is founded on the computational intractability of the above problems. The hardness of these problems implies that the discrete logarithm problems over $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are also intractable; and when selecting an elliptic curve the primary consideration is to ensure the discrete logarithm problems remain intractable on the selected curve.

5.3 Embedding degrees and secure curves

Let $\mathbb{G}$ be an N-order subgroup of the elliptic curve $E(F_q)$. The smallest positive integer k such that $N \mid q^k - 1$ is called the embedding degree of $\mathbb{G}$ relative to N, also known as the embedding degree of $E(F_q)$ relative to N.

Let $\mathbb{G}_1$ be an N-order subgroup of $E(F_{q^{d_1}})$, where $d_1 \mid k$, and $\mathbb{G}_2$ be an N-order subgroup of $E(F_{q^{d_2}})$, where $d_2 \mid k$, then the range $\mathbb{G}_T$ of the bilinear pairings based on the elliptic curves is a subgroup of $F^*_{q^k}$. Thus, the bilinear pairings based on the elliptic curves can convert the elliptic curve discrete logarithm problem to the discrete logarithm problem over the finite field $F^*_{q^k}$. The security of the curve improves as the size of the extension field increases (if no faster discrete logarithm algorithm exists in the field), yet it becomes harder to compute the bilinear pairings. Hence it is necessary to adopt an elliptic curve with an appropriate embedding degree while achieving the desired security level. This standard specifies that $q^k > 2^{1536}$.

This standard specifies the use of the following curves:
a) Ordinary curves whose base field is F_q, where q is a prime greater than 2^{191}, and the embedding degree $k = 2^i 3^j$, where $i > 0$ and $j \geq 0$.
b) Supersingular curves whose base field is F_q, where q is a prime greater than 2^{768}, and the embedding degree $k = 2$.

For N less than 2^{360}, it is recommended that
a) $N - 1$ has a prime factor greater than 2^{190}.
b) $N + 1$ has a prime factor greater than 2^{120}.

6 Data types and conversions

6.1 Data type

The data types include bit string, byte string, field element, elliptic curve point and integer in this standard.

Bit string: an ordered sequence of 0's and 1's.

Byte string: an ordered sequence of bytes, where one byte contains 8 bits and the leftmost bit is the most significant bit.

Field element: the elements of finite field F_{q^m} $(m \geq 1)$.

Elliptic curve point: a point $P \in E(F_{q^m})$ $(m \geq 1)$ is either a pair of field elements (x_P, y_P), where x_P, y_P satisfy the ecliptic curve equation, or the point at infinity O.

A point can be encoded as a byte string in many forms. A byte PC is used to indicate which form is used. The byte string representation of the point at infinity O is a unique zero byte $PC = 00$. A nonzero point $P = (x_P, y_P)$ can be represented as one of the following three byte string forms:

a) Compressed form, $PC = 02$ or 03;
b) Uncompressed form, $PC = 04$;
c) Hybrid form, $PC = 06$ or 07.

Note: The hybrid form contains both the compressed and uncompressed forms. In implementation, the hybrid form can be converted into the compressed or uncompressed forms. Implementation of the compressed and hybrid forms are optional in this standard. Please refer to Annex A.4 for the details of the compressed form.

6.2 Data type conversions

6.2.1 Conversion relations between data types

Figure 1 indicates the conversion relations between the data types. The subclauses for the

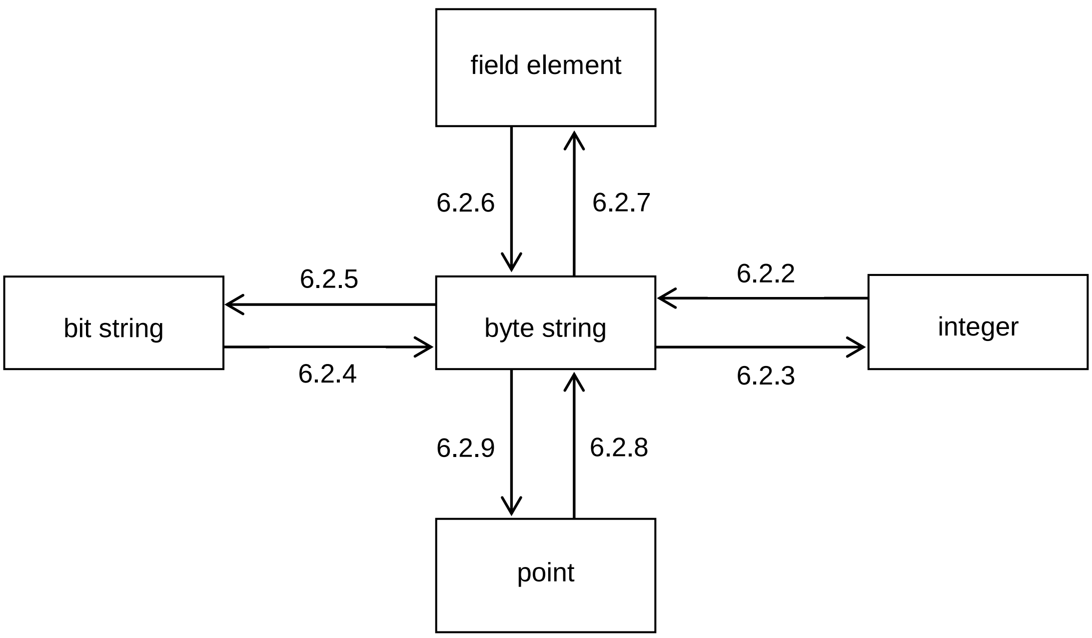

Figure 1: Data types and their conversions

corresponding conversion methods are given by the marks along the lines.

6.2.2 Conversion of an integer to a byte string

Input: a non-negative integer x, and the target length of the byte string l (where l satisfies $2^{8l} > x$).

Output: a byte string M of l bytes long.

a) Let $M_{l-1}, M_{l-2}, \dots, M_0$ be the individual bytes of M from left to right.

b) The bytes of M satisfy:

$$x = \sum_{i=0}^{l-1} 2^{8i} M_i.$$

6.2.3 Conversion of a byte string to an integer

Input: a byte string M of l bytes long.

Output: an integer x.

a) Let $M_{l-1}, M_{l-2}, \dots, M_0$ be the individual bytes of M from left to right.

b) Convert M to the integer x:

$$x = \sum_{i=0}^{l-1} 2^{8i} M_i.$$

6.2.4 Conversion of a bit string to a byte string

Input: a bit string s of n bits long.

Output: a byte string M of l bytes long, where $l = \lceil n/8 \rceil$.

a) Let $s_{n-1}, s_{n-2}, \dots, s_0$ be the individual bits of s from left to right.

b) Let $M_{l-1}, M_{l-2}, \dots, M_0$ be the individual bytes of M from left to right, then

$M_i = s_{8i+7} s_{8i+6} \dots s_{8i+1} s_{8i}$, where $0 \le i < l$, and when $8i + j \ge n$ and $0 < j \le 7$, $s_{8i+j} = 0$.

6.2.5 Conversion of a byte string to a bit string

Input: a byte string M of l bytes long.

Output: a bit string s of n bits long, where $n = 8l$.

a) Let $M_{l-1}, M_{l-2}, \dots, M_0$ be the individual bytes of M from left to right.

b) Let $s_{n-1}, s_{n-2}, \dots, s_0$ be the individual bits of s from left to right, then s_i is the $(i - 8j + 1)^{\text{th}}$ bit of M_j from the right, where $j = \lfloor i/8 \rfloor$.

6.2.6 Conversion of a field element to a byte string

Input: an element $\alpha = (\alpha_{m-1}, \alpha_{m-2}, \dots, \alpha_1, \alpha_0)$ in F_{q^m} $(m \geq 1)$, and $q = p$.

Output: a byte string s of length l, where $l = \lceil \log_2 q/8 \rceil \times m$.

a) If $m = 1$, then $\alpha = \alpha_0$ $(q = p)$, α is an integer in $[0, q-1]$, convert α to a byte string S of l bytes long as specified in 6.2.2.

b) If $m > 1$, then $\alpha = (\alpha_{m-1}, \alpha_{m-2}, \dots, \alpha_1, \alpha_0)$ $(q = p)$, where $\alpha_i \in F_q, i = 0,1, \dots, m-1$.

   1) Let $r = \lceil \log_2 q/8 \rceil$.

   2) For i from $m - 1$ to 0:
   Convert α_i $(q = p)$ to a byte string s_i of r bytes long as specified in 6.2.2.

   3) $S = s_{m-1} || s_{m-2} || \dots || s_0$.

6.2.7 Conversion of a byte string to a field element

Case 1: Convert to element in the base field

Input: a field F_q, $q = p$, and a byte string S of l bytes long, where $l = \lceil \log_2 q/8 \rceil$.

Output: an element α in F_q.

If $q = p$, convert S to an integer α as specified in 6.2.3. If α is not in the range $[0, q-1]$, report an error.

Case 2: Convert to element in extension field

Input: a field F_{q^m} $(m \geq 2)$, $q = p$, and a byte string S of l bytes long, where $l = \lceil \log_2 q/8 \rceil \times m$.

Output: an element α in F_{q^m}.

a) Equally divide the byte string S into m parts, where the length of each part is l/m bytes long, denote it as $S = (S_{m-1}, S_{m-2}, \dots, S_1, S_0)$.

b) For i from $m - 1$ to 0:

   Convert S_i to an integer α_i as specified in 6.2.3, and if α is not in $[0, q-1]$, report an error.

c) If $q = p$, output $a = (a_{m-1}, a_{m-2}, \dots, a_1, a_0)$.

6.2.8 Conversion of a point to a byte string

There are two cases in the conversion of a point to a byte string.

The first case is that in the computation process, convert the elliptic curve point to a byte string before setting it as the input of some function (e.g., a hash function). In this case, we only need to convert the point to byte string.

The second case is when transmitting or storing elliptic curve points, in order to reduce the transmission quantity or storage space, we can use the compressed or the hybrid compressed form of the points. In such case, we need to add an identifier PC to indicate the encoding form of the point.

The details of the two cases of conversion are as follows.

Case 1: **Direct conversion**

Input: a point $P = (x_P, y_P)$ on the elliptic curve $E(F_{q^m})$, where $P \neq O$.

Output: a byte string $X_1||Y_1$ of $2l$ bytes long. (If $m = 1$, $l = \lceil \log_2 q/8 \rceil$; if $m > 1$, $l = \lceil \log_2 q/8 \rceil \times m$.)

a) Convert the field element x_P to the byte string X_1 of l bytes long as specified in 6.2.6;

b) Convert the field element y_P to the byte string Y_1 of l bytes long as specified in 6.2.6;

c) Output the byte string $X_1||Y_1$.

Case 2: Conversion by adding a byte string identifier *PC*

Input: a point $P = (x_P, y_P)$ on the elliptic curve $E(F_{q^m})$, where $P \neq O$.

Output: a byte string PO. If the uncompressed form or the hybrid form is used, output a byte string of length $2l + 1$; if the compressed form is used, output a byte string of $l + 1$ bytes long. (If $m = 1$, $l = \lceil \log_2 q/8 \rceil$; if $m > 1$, $l = \lceil \log_2 q/8 \rceil \times m$.)

a) Convert the field element x_P to the byte string X_1 of l bytes long as specified in 6.2.6;

b) If the compressed form is used, then

   1) Compute the bit $\tilde{y}_P$. (See Annex A.4.)

   2) If $\tilde{y}_P = 0$, then let $PC = 02$; if $\tilde{y}_P$=1, $PC = 03$;

   3) Output the byte string $PO = PC||X_1$.

c) If the uncompressed form is used, then

   1) Convert the field element y_P to the byte string Y_1 of l bytes long as specified in 6.2.6;

   2) Let $PC = 04$;

   3) Output the byte string $PO = PC||\ X_1||Y_1$.

d) If the hybrid form is used, then

   1) Convert the field element y_P to the byte string Y_1 of l bytes long as specified in 6.2.6;

   2) Compute the bit $\tilde{y}_P$; (See Annex A.4.)

   3) If $\tilde{y}_P = 0$, then let $PC = 06$; if $\tilde{y}_P = 1$, $PC = 07$;

   4) Output the byte string $PO = PC||\ X_1||Y_1$.

6.2.9 Conversion of a byte string to a point

The conversion of a byte string to a point is the inverse process of 6.2.8. The conversion is explained in the following two cases.

Case 1: Direct conversion

Input: field elements a and b which define an elliptic curve over F_{q^m} $(m \geq 1)$, and the byte string $X_1||Y_1$ of length $2l$ bytes long. The lengths of both X_1 and Y_1 are l bytes. (If $m = 1$, $l = \lceil \log_2 q/8 \rceil$; if $m > 1$, $l = \lceil \log_2 q/8 \rceil \times m$.).

Output: a point $P = (x_P, y_P)$ of the elliptic curve, where $P \neq O$.

a) Convert the byte string X_1 to a field element x_P as specified in 6.2.7;

b) Convert the byte string Y_1 to a field element y_P as specified in 6.2.7;

Case 2: Conversion of a byte string containing the byte identifier *PC*

Input: field elements a and b which define an elliptic curve over F_{q^m} $(m \geq 1)$, and the byte string PO. If the uncompressed or hybrid forms are used, the length of PO is $2l + 1$ bytes long. If the compressed form is used, the length of PO is $l + 1$ bytes long. (If $m = 1$, then $l = \lceil \log_2 q/8 \rceil$; if $m > 1$, then $l = \lceil \log_2 q/8 \rceil \times m$.)

Output: a point $P = (x_P, y_P)$ of the elliptic curve, where $P \neq O$.

a) If the compressed form is used, then $PO = PC||X_1$; if the uncompressed or hybrid forms are used, $PO = PC||\, X_1||Y_1$, where PC is a single byte, and both X_1 and Y_1 are byte strings of l bytes long;

b) Convert the byte string X_1 to a field element x_P as specified in 6.2.7;

c) If the compressed form is used, then

   1) Check whether $PC = 02$ or $PC = 03$; if not, report an error;
   2) If $PC = 02$, then let $\tilde{y}_P = 0$; if $PC = 03$, let $\tilde{y}_P = 1$;
   3) Convert $(x_P, \tilde{y}_P)$ to a point (x_P, y_P) on the elliptic curve; (See Annex A.4.)

d) If the uncompressed form is used, then

   1) Check whether PC = 04; if not, report error;

   2) Convert the byte string Y_1 to a field element y_P as specified in 6.2.7;

e) If the hybrid form is used, then

   e.1) Check whether $PC = 06$ or $PC = 07$; if not, report an error;

   e.2) Perform e.2.1) or e.2.2):

   - Convert the byte string Y_1 to a field element y_P as specified in 6.2.7;

   - If $PC = 06$, then let $\tilde{y}_P = 0$, otherwise let $\tilde{y}_P = 1$; convert $(x_P, \tilde{y}_P)$ to a point (x_P, y_P) on the elliptic curve; (See Annex A.4.)

f) Check whether (x_P, y_P) satisfies the equation of the curve; if not, report an error;

g) $P = (x_P, y_P)$.

7 System parameters and parameters verification

7.1 System parameters

The system parameters include:

a) The curve identifier cid is denoted by one byte: 0x10 represents an ordinary curve over F_q (where the prime number $q > 3$), 0x11 represents a supersingular curve over F_q, and 0x12 represents an ordinary curve and the corresponding twisted curve over F_q;

b) The parameter of the base field F_q of the elliptic curve: the parameter of the base field is a prime number $q > 3$;

c) Two elements a and b in F_q, which define the equation of the elliptic curve $E: y^2 = x^3 + ax + b$; the twisted curve parameter β (if the least 4 significant bits of cid is 2);

d) The cofactor cf and a prime number N, where $cf \times N = \#E(F_q)$. GM/T 0044–2016 requires $N > 2^{191}$ and N is not divisible by cf. If $N < 2^{360}$, GM/T 0044–2016 recommends that $N - 1$ has prime factors greater than 2^{190} and $N + 1$ has prime factors greater than 2^{120};

e) The embedding degree k of the curve $E(F_q)$ relative to N. (The cyclic group with order $(\mathbb{G}_T, \cdot) \subset F^*_{q^k}$). GM/T 0044–2016 specifies that $q^k > 2^{1536}$;

f) A generator $P_1 = (x_{P_1}, y_{P_1})$ of the cyclic group $(\mathbb{G}_1, +)$, where $P_1 \neq O$;

g) A generator $P_2 = (x_{P_2}, y_{P_2})$ of the cyclic group $(\mathbb{G}_2, +)$, where $P_2 \neq O$;

h) The bilinear pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is denoted by one byte identifier eid: 0x01 represents the Tate pairing, 0x02 represents the Weil pairing, 0x03 represents the Ate pairing, and 0x04 represents the R-ate pairing;

i) (Optional) The parameters d_1, d_2, both of which are factors of k;

j) (Optional) The homomorphism Ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ such that $P_1 = \Psi(P_2)$;

k) (Optional) The characteristic of the base field of the BN curves, the order of curve r, and the trace of the Frobenius map which can be determined by the parameter t, where t is at least 63 bits long.

7.2 Verification of the system parameters

The following conditions shall be verified by the generator of the system parameters. They can also be verified by the users of the system parameters.

Input: the set of the system parameters.

Output: if all parameters are valid, output “valid”; otherwise output “invalid”.

a) Verify that q is a prime greater than 3; (See Annex C.1.5.)

b) Verify that a, b are integers in $[0, q-1]$;

c) Verify that $4a^3 + 27b^2 \neq 0$ over F_q; if the least 4 significant bits of cid are 2, verify that β is a non-square element; (See Annex C.1.4.3.1.)

d) Verify that N is a prime greater than 2^{191} and cf is not divisible by N; if $N < 2^{360}$, verify that $N-1$ has prime factors greater than 2^{190} and $N+1$ has prime factors greater than 2^{120};

e) Verify that $|q + 1 - cf \times N| < 2q^{1/2}$;

f) Verify that $q^k > 2^{1536}$ and k is the smallest positive integer m such that $N \mid (q^m - 1)$;

g) Verify that (x_{P_1}, y_{P_1}) is an element of $\mathbb{G}_1$;

h) Verify that (x_{P_2}, y_{P_2}) is an element of $\mathbb{G}_2$;

i) Verify $e(P_1, P_2) \in F_{q^k}^* \backslash \{1\}$, and $e(P_1, P_2)^N = 1$;

j) (Optional) Verify $d_1, d_2 \mid k$;

k) (Optional) Verify that $P_1 = \Psi(P_2)$;

If any of the above verification fails, output "invalid"; otherwise output "valid".

Annex A
(informative)

Elliptic curve basics

A.1 Finite field

A.1.1 Prime field F_p

Suppose p is prime, then in the set of remainders $\{0,1,2,\dots,p-1\}$ modulo p, the addition and multiplication in terms of the arithmetic of integers modulo p form a p-order prime field, which is symbolized by F_p. The additive identity is 0, while the multiplicative identity is 1. The elements of F_p have the following operation rules:

-- **Addition:** if $a, b \in F_p$, then $a + b = r$, where $r = (a + b) \bmod p$, $r \in [0, p - 1]$.
-- **Multiplication:** if $a, b \in F_p$, then $a \cdot b = s$, where $s = (a \cdot b) \bmod p$, $s \in [0, p - 1]$.

Let F_p^* be the multiplicative group composed of all nonzero elements of F_p. Since F_p^* is a multiplicative group, there is at least one element g in F_p, satisfying that any nonzero element in F_p can be represented by the power of g. We call g the generator (primitive element) of F_p^*, and $F_p^* = \{g^i \mid 0 \le i \le p-2\}$. Let $a = g^i \in F_p^*$, and $0 \le i \le p-2$, then the multiplicative inverse of a is: $a^{-1} = g^{p-1-i}$.

Example 1: the prime field $F_{19} = \{0, 1, 2, \dots, 18\}$.

Example of addition in F_{19}: $10, 14 \in F_{19}$, $10 + 14 = 24$, $24 \bmod 19 = 5$, then $10 + 14 = 5$.

Example of multiplication in F_{19}: $7, 8 \in F_{19}$, $7 \times 8 = 56$, $56 \bmod 19 = 18$, then $7 \cdot 8 = 18$.

13 is a generator of F_{19}^*, then the elements of F_{19}^* can be represented by the powers of 13:
$13^0 = 1, 13^1 = 13, 13^2 = 17, 13^3 = 12, 13^4 = 4, 13^5 = 14, 13^6 = 11, 13^7 = 10, 13^8 = 16, 13^9 = 18, 13^{10} = 6, 13^{11} = 2, 13^{12} = 7, 13^{13} = 15, 13^{14} = 5, 13^{15} = 8, 13^{16} = 9, 13^{17} = 3, 13^{18} = 1$.

A.1.2 Finite field F_{q^m}

Suppose q is a prime or a prime power, $f(x)$ be an m-degree $(m > 1)$ irreducible polynomial (which is called the reduced polynomial or the field polynomial) in the polynomial ring $F_q[x]$, the quotient ring $F_q[x]/(f(x))$ be a finite field composed of q^m elements, then F_{q^m} is an extension field of F_q, F_q is a subfield of F_{q^m}, m is the extension degree. F_{q^m} can be seen as the m-dimensional vector space of F_q, that is to say there exist m elements $\alpha_0, \alpha_1, \dots, \alpha_{m-1}$ in F_{q^m}, such that $\forall \alpha \in F_{q^m}$, α can be uniquely represented by $\alpha = a_{m-1}\alpha_{m-1} + \cdots + a_0\alpha_0 + a_1\alpha_1$ $(a_i \in F_q)$, then $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ is called a basis of F_{q^m} over F_q. Given such a basis, then we can use the vector $(a_0, a_1, \dots, a_{m-1})$ to represent the field element α.

There are many possible choices for the selection of a basis, such as the polynomial basis and the normal basis.

Suppose the irreducible polynomial $f(x)$ is a monic polynomial $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0$ $(f_i \in F_q, i = 0,1,\dots,m-1)$, and the elements of F_{q^m} can be represented by all polynomials with degree less than m in the polynomial ring $F_q[x]$, that is, $F_{q^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 \mid a_i \in F_q, i = 0,1,\dots,m-1\}$. The set of polynomials $\{x^{m-1}, x^{m-2}, \dots, x, 1\}$ is a basis of F_{q^m} as a vector space over F_q, which is called a polynomial basis. When m has a divisor d $(1 < d < m)$, F_{q^d} could be extended to F_{q^m}. If a suitable m/d-degree irreducible polynomial is selected from $F_{q^d}[x]$ to act as F_{q^m}'s reduced polynomial on F_{q^d}, then F_{q^m} could be generated according to the towering method. This extension's basic forms are still vectors composed of the elements of F_q. For example, when $m = 6$, F_q could be extended three times to the extension field F_{q^3}, and F_{q^3} could be further extended twice to the extension field F_{q^6}. F_q could be extended twice to the extension field F_{q^2}, and F_{q^2} could be further extended three times to the extension field F_{q^6}.

The basis of the form $\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}\}$ of F_{q^m} over F_q are called normal basis, where $\beta \in F_{q^m}$. $\forall a \in F_{q^m}$, a could be represented as $a = a_0\beta + a_1\beta^q + \cdots + a_{m-1}\beta^{q^{m-1}}$, where $a_i \in F_q$, $i = 0,1,\ldots, m-1$. For any finite field F_q and its extension field F_{q^m}, such basis always exist.

Unless otherwise specified, all elements in F_{q^m} are represented by the polynomial basis.

The field element $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ could be represented by the vector $(a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$ in terms of the polynomial basis, so $F_{q^m} = \{(a_{m-1}, a_{m-2}, \ldots, a_1, a_0) \mid a_i \in F_q, i = 0,1,\ldots, m-1\}$.

The multiplicative identity is represented by $(0, \ldots, 0, 1)$, and the zero element is represented by $(0, \ldots, 0, 0)$. The addition and multiplication of the field elements are defined as follows.

Addition. $\forall (a_{m-1}, a_{m-2}, \ldots, a_1, a_0), (b_{m-1}, b_{m-2}, \ldots, b_1, b_0) \in F_{q^m}$, then $(a_{m-1}, a_{m-2}, \ldots, a_1, a_0) + (b_{m-1}, b_{m-2}, \ldots, b_1, b_0) = (c_{m-1}, c_{m-2}, \ldots, c_1, c_0)$, where $c_i = a_i + b_i$, $i = 0,1, \ldots, m-1$. That is, addition is implemented by component-wise addition in F_q.

Multiplication. $\forall (a_{m-1}, a_{m-2}, \ldots, a_1, a_0), (b_{m-1}, b_{m-2}, \ldots, b_1, b_0) \in F_{q^m}$, then $(a_{m-1}, a_{m-2}, \ldots, a_1, a_0) \cdot (b_{m-1}, b_{m-2}, \ldots, b_1, b_0) = (r_{m-1}, r_{m-2}, \ldots, r_1, r_0)$, where the polynomial $r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \cdots + r_1x + r_0$ is the remainder of $(a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0) \cdot (b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0)$ modulo $f(x)$ in $F_q[x]$.

F_{q^m} contains q^m elements. Let $F_{q^m}^*$ be the multiplicative group composed of all nonzero elements in F_{q^m}. Since F_{q^m} is a multiplicative group, there exists at least one element g in F_{q^m} such that any nonzero element of F_{q^m} can be represented by the powers of g. g is called the generator (or primitive element) of $F_{q^m}^*$, and $F_{q^m}^* = \{g^i \mid 0 \le i \le q^m - 2\}$. Let $a = g^i \in F_{q^m}^*$, where $0 \le i \le q^m - 2$, then the multiplicative inverse of a is $a^{-1} = g^{q^m - 1 - i}$.

Example 2: the polynomial basis representation of F_{3^2}.

Let $f(x) = x^2 + 1$ be an irreducible polynomial over F_3, then the elements of F_{3^2} are:
(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2).
Addition: $(2,1) + (2,0) = (1,1)$.
Multiplication: $(2,1) \cdot (2,0) = (2,2)$

$$\begin{aligned}(2x+1)\cdot 2x &= 4x^2 + 2x \\ &= x^2 + 2x \\ &= 2x + 2 \ (\mathrm{mod}\ f(x))\end{aligned}$$

That is, $2x + 2$ is the reminder of $(2x + 1) \cdot 2x$ modulo $f(x)$.

The multiplicative identity is $(0, 1)$, and $\alpha = x + 1$ is a generator of $F_{3^2}^*$, then the powers of α are
$\alpha^0 = (0,1)$, $\alpha^1 = (1,1)$, $\alpha^2 = (2,0)$, $\alpha^3 = (2,1)$, $\alpha^4 = (0,2)$, $\alpha^5 = (2,2)$, $\alpha^6 = (1,0)$, $\alpha^7 = (1,2)$, $\alpha^8 = (0,1)$.

A.1.3 Elliptic curves over finite fields

A.1.3.1 Overview

There are two common representations for the elliptic curves over finite fields: an affine coordinate and a projective coordinate.

A.1.3.2 Affine coordinate

Suppose p is a prime greater than 3, the elliptic curve equation over F_{p^m} in the affine coordinate system can be simplified as $y^2 = x^3 + ax + b$, where $a, b \in F_p$, satisfying $(4a^3 + 27b^2) \bmod p \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{p^m}) = \{(x, y) \mid x, y \in F_{p^m}$, satisfying the equation $y^2 = x^3 + ax + b\} \cup \{O\}$, where O in the point at infinity, also called the zero point.

The points on $E\left(F_{p^m}\right)$ form an abelian group according to the following addition operation rules:

a) $O + O = O$;

b) $\forall P = (x, y) \in E(F_{p^m})\backslash\{O\}, P + O = O + P = P$;

c) $\forall P = (x, y) \in E(F_{p^m})\backslash\{O\}$, the inverse element of P is $-P = (x, -y), P + (-P) = O$;

d) $P_1 = (x_1, y_1) \in E(F_{p^m})\backslash\{O\}, P_2 = (x_2, y_2) \in E(F_{p^m})\backslash\{O\}$, and $P_3 = (x_3, y_3) = P_1 + P_2 \neq O$, then

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases}$$

where

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2, \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } x_1 = x_2, \text{and } P_2 \neq -P_1. \end{cases}$$

Example 3: an elliptic curve over F_{19}

The equation over F_{19}: $y^2 = x^3 + x + 1$, where $a = 1, b = 1$. The points on the curve are:

(0,1), (0,18), (2,7), (2,12), (5,6), (5,13), (7,3), (7,16), (9,6), (9,13), (10,2), (10,17), (13,8), (13,11), (14,2), (14,17), (15,3), (15,16), (16,3), (16,16).

There are 21 points (including O) on $E(F_{19})$.

a) Let $P_1 = (10, 2), P_2 = (9, 6)$, then compute $P_3 = P_1 + P_2$:

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{6 - 2}{9 - 10} = \frac{4}{-1} = -4 \equiv 15 \pmod{19}, \\ x_3 &= 152 - 10 - 9 = 225 - 10 - 9 = 16 - 10 - 9 = -3 \equiv 16 \pmod{19}, \\ y_3 &= 15 \times (10 - 16) - 2 = 15 \times (-6) - 2 \equiv 3 \pmod{19}, \end{aligned}$$

thus, $P_3 = (16, 3)$.

b) Let $P_1 = (10, 2)$, then compute $[2]P_1$:

$$\begin{aligned} \lambda &= \frac{3x_1^2 + a}{2y_1} = \frac{3 \times 10^2 + 1}{2 \times 2} = \frac{3 \times 5 + 1}{4} = \frac{16}{4} = 4 \pmod{19}, \\ x_3 &= 42 - 10 - 10 = -4 \equiv 15 \pmod{19}, \end{aligned}$$

$y_3 = 4 \times (10 - 15) - 2 = -22 \equiv 16 (\mathrm{mod}\ 19)$,

thus, $[2]P_1 = (15, 16)$.

A.1.3.3 Projective coordinate

A.1.3.3.1 Standard projective coordinate system

The elliptic curve equation over F_{p^m} in the standard projective coordinate system can be simplified as $y^2 z = x^3 + axz^2 + bz^3$, where $a, b \in F_{p^m}$, satisfying $4a^3 + 27b^2 \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{p^m}) = \{(x, y, z) \mid x, y, z \in F_{p^m},$ satisfying the equation $y^2 z = x^3 + axz^2 + bz^3\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_{p^m}$ $(u \neq 0)$ such that $x_1 = ux_2$, $y_1 = uy_2$, and $z_1 = uz_2$, then these two triples are equivalent, and they represent the same point.

If $z \neq 0$, let $X = x/z$, $Y = y/z$, then the standard projective coordinates can be converted to the affine coordinates: $Y^2 = X^3 + aX + b$.

If $z = 0$, then the point (0,1,0) corresponds to the point at infinity O of the affine coordinate system.

In the standard projective coordinate system, the addition of the points on $E(F_{p^m})$ is defined as follows:
a) $O + O = O$;
b) $\forall P = (x, y, z) \in E(F_{p^m}) \backslash \{O\}$, $P + O = O + P = P$;
c) $\forall P = (x, y, z) \in E(F_{p^m}) \backslash \{O\}$, the inverse element of P is $-P = (ux, -uy, uz)$, $u \in F_{p^m}$ $(u \neq 0)$, and $P + (-P) = O$;
d) Let $P_1 = (x_1, y_1, z_1) \in E(F_{p^m}) \backslash \{O\}$, $P_2 = (x_2, y_2, z_2) \in E(F_{p^m}) \backslash \{O\}$, and $P_3 = P_1 + P_2 = (x_3, y_3, z_3) \neq O$.
If $P_1 \neq P_2$, then
$\lambda_1 = x_1 z_2$, $\lambda_2 = x_2 z_1$, $\lambda_3 = \lambda_1 - \lambda_2$, $\lambda_4 = y_1 z_2$, $\lambda_5 = y_2 z_1$, $\lambda_6 = \lambda_4 - \lambda_5$, $\lambda_7 = \lambda_1 + \lambda_2$, $\lambda_8 = z_1 z_2$, $\lambda_9 = \lambda_3^2$, $\lambda_{10} = \lambda_3 \lambda_9$, $\lambda_{11} = \lambda_8 \lambda_6^2 - \lambda_7 \lambda_9$, $x_3 = \lambda_3 \lambda_{11}$, $y_3 = \lambda_6 (\lambda_9 \lambda_1 - \lambda_{11}) - \lambda_4 \lambda_{10}$, $z_3 = \lambda_{10} \lambda_8$.
If $P_1 = P_2$, then
$\lambda_1 = 3x_1^2 + az_1^2$, $\lambda_2 = 2y_1 z_1$, $\lambda_3 = y_1^2$, $\lambda_4 = \lambda_3 x_1 z_1$, $\lambda_5 = \lambda_2^2$, $\lambda_6 = \lambda_1^2 - 8\lambda_4$, $x_3 = \lambda_2 \lambda_6$, $y_3 = \lambda_1 (4\lambda_4 - \lambda_6) - 2\lambda_5 \lambda_3$, $z_3 = \lambda_2 \lambda_5$.

A.1.3.3.2 Jacobian projective coordinate system

The elliptic curve equation over F_{p^m} in the Jacobian projective coordinate system can be simplified as $y^2 = x^3 + axz^4 + bz^6$, where $a, b \in F_{p^m}$, satisfying $4a^3 + 27b^2 \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{p^m}) = \{(x, y, z) \mid x, y, z \in F_{p^m}$, satisfying the equation $y^2 = x^3 + axz^4 + bz^6\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_{p^m}$ $(u \neq 0)$ such that $x_1 = u^2 x_2$, $y_1 = u^3 y_2$, and $z_1 = uz_2$, then these two triples are equivalent, and they represent the same point.

If $z \neq 0$, let $X = x/z^2$, $Y = y/z^3$, then the Jacobian projective coordinates can be converted to the affine coordinates: $Y^2 = X^3 + aX + b$.

If $z = 0$, then the point (1,1,0) corresponds to the point at infinity O of the affine coordinate system.

In the Jacobian projective coordinate system, the addition of the points on $E(F_{p^m})$ is defined as follows:
a) $O + O = O$;
b) $\forall P = (x, y, z) \in E(F_{p^m}) \backslash \{O\}$, $P + O = O + P = P$;
c) $\forall P = (x, y, z) \in E(F_{p^m}) \backslash \{O\}$, the inverse element of P is $-P = (u^2 x, -u^3 y, uz)$, $u \in F_{p^m}$ $(u \neq 0)$, and $P + (-P) = O$;
d) Let $P_1 = (x_1, y_1, z_1) \in E(F_{p^m}) \backslash \{O\}$, $P_2 = (x_2, y_2, z_2) \in E(F_{p^m}) \backslash \{O\}$, and $P_3 = P_1 + P_2 =$

$(x_3, y_3, z_3) \neq O$.
If $P_1 \neq P_2$, then
$\lambda_1 = x_1 z_2^2, \lambda_2 = x_2 z_1^2, \lambda_3 = \lambda_1 - \lambda_2, \lambda_4 = y_1 z_2^3, \lambda_5 = y_2 z_1^3, \lambda_6 = \lambda_4 - \lambda_5, \lambda_7 = \lambda_1 + \lambda_2, \lambda_8 = \lambda_4 + \lambda_5, \lambda_9 = \lambda_7 \lambda_3^2, x_3 = \lambda_6^2 - \lambda_9, \lambda_{10} = \lambda_9^2 - 2x_3, y_3 = (\lambda_{10}\lambda_6 - \lambda_8 \lambda_3^3)/2, z_3 = z_1 z_2 \lambda_3$.
If $P_1 = P_2$, then
$\lambda_1 = 3x_1^2 + a z_1^4, \lambda_2 = 4x_1 y_1^2, \lambda_3 = 8y_1^4, x_3 = \lambda_1^2 - 2\lambda_2, y_3 = \lambda_1(\lambda_2 - x_3) - \lambda_3, z_3 = 2y_1 z_1$.

A.1.4 Order of elliptic curves over finite field

The order of an elliptic curve over finite field F_{q^m} is the number of elements in the set $E(F_{q^m})$, denoted by $\#E(F_{q^m})$. According to the Hasse theorem, we have $q^m + 1 - 2q^{m/2} \leq \#E(F_{q^m}) \leq q^m + 1 + 2q^{m/2}$, that is to say, $\#E(F_{q^m}) = q^m + 1 - t$, where t is called the Frobenius trace, satisfying $|t| \leq 2q^{m/2}$.

If the Frobenius trace t is divisible by the characteristic of F_{q^m}, this curve is supersingular; otherwise, it is non-supersingular.

Suppose $E(F_{q^m})$ is an elliptic curve over F_{q^m}, the integer r and q^m are coprime, then the r-order twisted subgroup of $E(F_{q^m})$ is $E(F_{q^m})[r] = \{P \in E(F_{q^m}) \mid [r]P = O\}$ and any $P \in E(F_{q^m})[r]$ is an r-fulcrum.

A.2 Elliptic curve scalar multiplication

The operation of adding a point along an elliptic curve to itself repeatedly is called the scalar multiplication of the point. Let u be a positive integer, P be a point on an elliptic curve, then the u multiple of the point P is denoted as $Q = [u]P = \underbrace{P + P + \cdots + P}_{u\ P\prime s}$.

Scalar multiplication can be extended to 0-scalar and negative-scalar: $[0]P = 0$, $[-u]P = [u](-P)$.

There are many ways to implement elliptic curve scalar multiplication, and the most fundamental three methods are noted here, where $1 \leq u < N$.

Algorithm 1: Binary expansion method

Input: a point P, an l-bit long integer $u = \sum_{j=0}^{l-1} u_j 2^j, u_j \in \{0, 1\}$.

Output: $Q = [u]P$.

a) Set $Q = O$;

b) For $j = l - 1$ to 0:

b.1) $Q = [2]Q$;

b.2) If $u_j = 1$, then $Q = Q + P$;

c) Output Q.

Algorithm 2: Addition and subtraction method

Input: a point P, an l-bit long integer $u = \sum_{j=0}^{l-1} u_j 2^j, u_j \in \{0, 1\}$.

Output: $Q = [u]P$.

a) Suppose the binary representation of $3u$ is $h_r h_{r-1} \dots h_1 h_0$, and the most significant bit h_r is 1. Obviously $r = l$ or $r = l + 1$;

b) The binary representation of u is $u_r u_{r-1} \dots u_1 u_0$;

c) Set $Q = P$;

d) For $i = r - 1$ to 1:

d.1) $Q = [2]Q$;

d.2) If $h_i = 1$ and $u_i = 0$, then $Q = Q + P$;

d.3) If $h_i = 0$ and $u_i = 1$, then $Q = Q - P$;

e) Output Q.

Note: Subtracting the point (x, y) is equivalent to adding the point $(x, -y)$. There are many different methods to accelerate this operation.

Algorithm 3: Sliding window method

Input: a point P, an l-bit long integer $u = \sum_{j=0}^{l-1} u_j 2^j, u_j \in \{0, 1\}$.

Output: $Q = [u]P$.

Let the window length $r > 1$.

Pre-computation:

a) $P_1 = P, P_2 = [2]P$;

b) For $i = 1$ to $2^{r-1} - 1$, compute $P_{2i+1} = P_{2i-1} + P_2$;

c) Set $j = l - 1, Q = 0$.

Main loop:

d) When $j \geq 0$:

d.1) if $u_j = 0$, then $Q = [2]Q, j = j - 1$;

d.2) otherwise

d.2.1) let t be the smallest integer satisfying $j - t + 1 \leq r$ and $u_t = 1$;

d.2.2) $h_j = \sum_{i=0}^{j-t} u_{t+i} 2^i$;

d.2.3) $Q = [2^{j-t+1}]Q + P_{h_j}$;

d.2.4) set $j = t - 1$;

e) Output Q.

A.3 Discrete logarithm problem

A.3.1 Methods to solve the field discrete logarithm problem

Let F_q^* be the multiplicative group composed of all nonzero elements in the finite field F_q. We call g the generator of F_q^*, and $F_q^* = \{g_i \mid 0 \leq i \leq q-2\}$. The order of $a \in F_q$ is the smallest positive integer t satisfying $a^t = 1$. The order of the multiplicative group F_q^* is $q-1$, so $t \mid q-1$.

Suppose the generator of the multiplicative group F_q^* is g and $y \in F_q^*$, the finite field discrete logarithm problem is to determine the integer $x \in [0, q-2]$ such that $y = g^x \bmod q$.

The existing attacks on the finite field discrete logarithm problem are:

a) Pohlig-Hellman method: let l be the largest prime divisor of $q-1$, then the time complexity is $O(l^{1/2})$;

b) BSGS method: the time and space complexity are both $(\pi q/2)^{1/2}$;

c) Pollard's method: the time complexity is $(\pi q/2)^{1/2}$;

d) Parallel Pollard's method: let s be the number of parallel processors, the time complexity reduces to $(\pi q/2)^{1/2}/s$;

e) Linear sieve method (for the prime fields F_q): the time complexity is $\exp((1+o(1))(\log q)^{1/2}(\log\log q)^{1/2})$;

f) Gauss integer method (for the prime fields F_q): the time complexity is $\exp((1+o(1))(\log q)^{1/2}(\log\log q)^{1/2})$;

g) Remainder listing sieve method (for prime fields F_q): the time complexity is $\exp((1+o(1))(\log q)^{1/2}(\log\log q)^{1/2})$;

h) Number field sieve method (for prime fields F_q): the time complexity is $\exp(\left((64/9)^{1/3}+o(1)\right)(\log q(\log\log q)^2)^{1/3})$;

i) Function field sieve method (for fields of small characteristics): the time complexity is $\exp(c(\log q(\log\log q)^2)^{1/4+o(1)})$ and quasi-polynomial time.

From the above enumerated methods for the finite field discrete logarithm problems and their time complexity, we know that: for discrete logarithm problems over fields of large characteristics, there are attack methods with sub-exponential complexity; for discrete logarithm problems over fields of small characteristics, there are quasi-polynomial time attack methods.

A.3.2 Methods to solve the elliptic curve discrete logarithm problem

For an elliptic curve $E(F_q)$, the point $P \in E(F_q)$ with order n and $Q \in \langle P \rangle$, the elliptic curve discrete logarithm problem is to determine the integer $u \in [0, n-1]$ such that $Q = [u]P$.

The existing attacks on ECDLP are:

a) Pohlig-Hellman method: let l be the largest prime divisor of n, then the time complexity is $O(l^{1/2})$;

b) BSGS method: the time and space complexity are both $(\pi n/2)^{1/2}$;

c) Pollard's method: the time complexity is $(\pi n/2)^{1/2}$;

d) Parallel Pollard's method: let r be the numbers of parallel processors, the time complexity reduces to $(\pi n/2)^{1/2}/r$;

e) MOV method: Reduces the ECDLP over supersingular curves and similar curves to DLP over F_q's small extension fields (This is a method of sub-exponential complexity);

f) Anomalous method: efficient attack methods for the anomalous curves (curves of $\#E(F_q) = q$) (This is a method of polynomial complexity);

g) GHS method: use Weil descent technique to solve the ECDLP of curves over binary extension field (the extension degree is a composite number), and convert the ECDLP to hyper-elliptic curve discrete logarithm problem, and there is the algorithm with sub-exponential complexity to this problem.

h) DGS-points decomposing method: use to compute the indexes used by elliptic curve discrete logarithm over low-degree extension fields. In some special cases, its complexity is lower than the square-root time method.

From the above description and analysis of ECDLP solutions and their time complexity, we can know that: for the discrete logarithm problem of general curves, the current solutions have exponential complexity, and no efficient attack method with sub-exponential complexity has been found; and for the discrete logarithm problem of some special curves, there are attack algorithms with polynomial complexity or sub-exponential complexity.

A.4 Compression of points on elliptic curve

A.4.1 Overview

For any nonzero point $P = (x_P, y_P)$ on $E(F_q)$, this point can be represented simply by the x-coordinate and a specific bit derived from y_P. This is the compression representation of points.

A4.2 Compression and decompression methods for points on elliptic curves over F_p

Let $P = (x_P, y_P)$ be a point on $E(F_p)$: $y^2 = x^3 + ax + b$, and $\tilde{y}_P$ be the rightmost bit of y_P, then P can be represented by x_P and the bit $\tilde{y}_P$.

The method of recovering y_P from x_P and $\tilde{y}_P$ is as follows:

a) Compute the field element $\alpha = x_P^3 + ax_P + b$ in F_p;

b) Compute the square root β of α in F_p (referring to Annex C.1.4). If no square root exists, then report an error;

c) If the rightmost bit of β is equal to $\tilde{y}_P$, then set $y_P = \beta$; otherwise set $y_P = p - \beta$.

A.4.3 Compression and decompression methods for points on elliptic curve over F_{q^m} (where q is an odd prime number and $m \geq 2$)

Let $P = (x_P, y_P)$ be a point on $E(F_{q^m}): y^2 = x^3 + ax + b$, then y_P can be represented as $(y_{m-1}, y_{m-2}, \dots, y_1, y_0)$; let $\tilde{y}_P$ be the rightmost bit of y_P, then P can be represented by x_P and the bit $\tilde{y}_P$.

The method of recovering y_P from x_P and $\tilde{y}_P$ is as follows:

a) Compute the field element $\alpha = x_P^3 + ax_P + b$ in F_{q^m} ;

b) Compute the square root β of α in F_{q^m} (referring to Annex C.1.4). If no square root exists, then report an error;

If in the representation $(\beta_{m-1}, \beta_{m-2}, \dots, \beta_1, \beta_0)$ of β, the rightmost bit of β_0 is equal to $\tilde{y}_P$, then set $y_P = \beta$; otherwise set $y_P = (\beta'_{m-1}, \beta'_{m-2}, \dots, \beta'_1, \beta'_0)$, where $\beta'_i = (q - \beta_i) \in F_q$, $i = 0,1, \dots, m-1$.

Annex B
(informative)

Computation of bilinear pairings over elliptic curves

B.1 Overview

Let an elliptic curve over finite field be $E(F_q)$. If $\#E(F_q) = cf \times r$, r is prime, cf is the cofactor, then the smallest positive integer k satisfying $r \mid q^k - 1$ is known as the elliptic curve's embedding degree relative to r. If $\mathbb{G}$ is an r order subgroup of $E(F_q)$, the embedding degree of $\mathbb{G}$ is k as well.

Let $\bar{F}_q$ be an algebraic closure of finite field F_q, and $E[r]$ the set of all points of order r in $E(\bar{F}_q)$.

B.2 Miller's algorithm

Let the equation of elliptic curves $E(F_{q^k})$ over F_{q^k} be $y^2 = x^3 + ax + b$, and define the straight line passing through the points U and V on $E(F_{q^k})$ as $g_{U,V}: E(F_{q^k}) \to F_{q^k}$. If the equation of the line passing through the points U and V is $\lambda x + \delta y + t = 0$, then set function $g_{U,V}(Q) = \lambda x_Q + \delta y_Q + t$, where $Q = (x_Q, y_Q)$. When $U = V$, $g_{U,V}$ is defined as the tangent line passing through the point U; if either U or V is the point at infinity, $g_{U,V}$ is a straight line perpendicular to the x-axis and passing through the other point. Generally, $g_{U,-U}$ is abbreviated as g_U.

Let $U = (x_U, y_U)$, $V = (x_V, y_V)$, $Q = (x_Q, y_Q)$, $\lambda_1 = (3x_V^2 + a)/(2y_V)$, $\lambda_2 = (y_U - y_V)/(x_U - x_V)$, then there should have the following properties:

a) $g_{U,V}(O) = g_{U,O}(Q) = g_{O,V}(Q) = 1$;

b) $g_{V,V}(Q) = \lambda_1(x_Q - x_V) - y_Q + y_V, Q \neq O$;

c) $g_{U,V}(Q) = \lambda_2(x_Q - x_V) - y_Q + y_V, Q \neq O, U \neq \pm V$;

d) $g_{V,-V}(Q) = x_Q - x_V, Q \neq O$.

Miller's algorithm is an efficient algorithm to compute bilinear pairings.

Miller's algorithm

Input: a curve E, two points P and Q on E, and an integer c.

Output: $f_{P,c}(Q)$.

a) The binary representation of c is $c_j \ldots c_1 c_0$, and the most significant bit c_j is 1;

b) Set $f = 1$, and $V = P$;

c) For $i = j - 1$ to 0:

  c.1) Compute $f = f^2 \cdot g_{V,V}(Q)/g_{2V}(Q)$, $V = [2]V$;

  c.2) If $c_i = 1$, let $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q)$, $V = V + P$.

d) Output f.

Generally, $f_{P,c}(Q)$ is known as the Miller function.

B.3 Computation of the Weil pairing

Let E be an elliptic curve over F_q, and r be a positive integer coprime to q. Suppose μ_r is the set of rth unit roots, and k is the embedding degree relative to r, that is $r \mid q^k - 1$, then $\mu_r \subset F_{q^k}$.

Let $\mathbb{G}_1 = E[r]$, $\mathbb{G}_2 = E[r]$, $\mathbb{G}_T = \mu_r$, then the Weil pairing is a bilinear mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, which is denoted as e_r.

Let $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, if $P = O$ or $Q = O$, then $e_r(P, Q) = 1$; if $P \neq O$ and $Q \neq O$, for randomly selected points $T \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, which are not the point at infinity, such that neither $P + T$ nor T equal to U or $U + Q$, then the Weil pairing is

$$e_r(P, Q) = \frac{f_{P+T,r}(Q + U) f_{T,r}(U) f_{U,r}(P + T) f_{Q+U,r}(T)}{f_{T,r}(Q + U) f_{P+T,r}(U) f_{Q+U,r}(P + T) f_{U,r}(T)}.$$

$f_{P+T,r}(Q + U)$, $f_{T,r}(Q + U)$, $f_{P+T,r}(U)$, $f_{T,r}(U)$, $f_{Q+U,r}(P + T)$, $f_{Q+U,r}(T)$, $f_{U,r}(P + T)$, $f_{U,r}(T)$ can be computed using the Miller algorithm. If the denominator happens to be 0 during computation, replace the point T or U and recompute.

B.4 Computation of the Tate pairing

Let E be an elliptic curve over F_q, r be a positive integer coprime to q, and k the embedding degree relative to r. Let Q be the r order on $E(F_{q^k})[r]$, and $\langle Q \rangle$ is the cyclic group generated by Q. $\left(F_{q^k}^*\right)^r$ is the set of the rth power of each element in $F_{q^k}^*$, $\left(F_{q^k}^*\right)^r$ is a subgroup of $F_{q^k}^*$, the quotient group of $F_{q^k}^*$ about $\left(F_{q^k}^*\right)^r$ is written as $F_{q^k}^* / \left(F_{q^k}^*\right)^r$.

Let $\mathbb{G}_1 = E(F_q)[r]$, $\mathbb{G}_2 = \langle Q \rangle$, $\mathbb{G}_T = F_{q^k}^* / \left(F_{q^k}^*\right)^r$, then the Tate pairing is a bilinear mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, written as t_r.

Let $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, if $P = O$ or $Q = O$, then $t_r = 1$; if $P \neq O$ and $Q \neq O$, for randomly selected point $U \in E\left(F_{q^k}\right)$ which is not the point at infinity, such that $P \neq Q$, $P \neq Q + U$, $U \neq -Q$, then the Tate pairing is

$$t_r(P, Q) = \frac{f_{P,r}(Q + U)}{f_{P,r}(U)}.$$

$f_{P,r}(Q + U)$ and $f_{P,r}(U)$ can be computed using the Miller algorithm. During the computation, if the denominator happens to be 0, replace the point U and re-compute.

In practice, the reduced Tate pairings as follows is generally used:

$$t_r(P, Q) = \begin{cases} f_{P,r}(Q)^{\frac{q^k - 1}{r}}, & Q \neq O, \\ 1, & Q = O. \end{cases}$$

The computation amount would be cut in half if the reduced Tate pairings is applied instead of the general Tate pairings. If the embedding degree k relative to r is an even number, then the computation method of reduced Tate pairings could be further optimized. Algorithm 1 describes the common methods applied to reduce Tate pairings, Algorithm 2, 3 and 4 deal with circumstances when $k = 2d$.

Algorithm 1

Input: an integer r coprime to q, $P \in E(F_q)[r]$, $Q \in E(F_{q^k})[r]$.

Output: $t_r(P, Q)$.

a) The binary representation of r is $r_j \ldots r_1 r_0$, and the most significant bit r_j is 1;

b) Set $f = 1, V = P$;

c) For $i = j - 1$ to 0:

c.1) Compute $f = f^2 \cdot g_{V,V}(Q)/g_{2V}(Q), V = [2]V$;

c.2) If $r_i = 1$, let $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q)$, $V = V + P$.

d) Compute $f = f^{q^d - 1}$;

e) Compute $f = f^{(q^d+1)/r}$.

f) Output f.

Algorithm 2

Input: an integer r coprime to q, $P \in E(F_q)[r]$, $Q \in E(F_{q^k})[r]$.

Output: $t_r(P, Q)$.

a) The binary representation of r is $r_j \ldots r_1 r_0$, and the most significant bit r_j is 1;

b) Set $f = 1, V = P$;

c) For $i = j - 1$ to 0:

c.1) Compute $f = f^2 \cdot g_{V,V}(Q)/g_{2V}(Q), V = [2]V$;

c.2) If $r_i = 1$, let $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q)$, $V = V + P$.

d) Compute $f = f^{q^d - 1}$;

e) Compute $f = f^{(q^d+1)/r}$;

f) Output f.

Algorithm 3

If F_{q^k} $(k = 2d)$ is seen as the quadratic extension of F_{q^d}, then the elements in F_{q^k} can be represented as $w = w_0 + iw_1$, where $w_0, w_1 \in F_{q^d}$, then the conjugate of w is $\overline{w} = w_0 - iw_1$, and in this case, the inverse in algorithm 1 can be replaced with conjugate.

Input: an integer r coprime to q, $P \in E(F_q)[r]$, $Q \in E(F_{q^k})[r]$.

Output: $t_r(P, Q)$.

a) The binary representation of r is $r_j \dots r_1 r_0$, and the most significant bit r_j is 1;

b) Set $f = 1, V = P$;

c) For $i = j - 1$ to 0:

c.1) Compute $f = f^2 \cdot g_{V,V}(Q)/g_{2V}(Q), V = [2]V$;

c.2) If $r_i = 1$, let $f = f \cdot g_{V,P}(Q)/\bar{g}_{V+P}(Q)$, $V = V + P$.

d) Compute $f = f^{q^d-1}$;

e) Compute $f = f^{(q^d+1)/r}$;

f) Output f.

Algorithm 4

When q is a prime greater than 3, then the point $Q \in E'$, where E' is the torsion curve of E. In this case, the algorithm could be further optimized.

Input: $P \in E(F_q)[r]$, $Q \in E'(F_{q^d})[r]$, an integer r.

Output: $t_r(P, Q)$.

a) The binary representation of r is $r_j \dots r_1 r_0$, and the most significant bit r_j is 1;

b) Set $f = 1, V = P$;

c) For $i = j - 1$ to 0:

c.1) Compute $f = f^2 \cdot g_{V,V}(Q), V = [2]V$;

c.2) If $r_i = 1$, let $f = f \cdot g_{V,P}(Q), V = V + P$.

d) Compute $f = f^{q^d-1}$;

e) Compute $f = f^{(q^d+1)/r}$;

f) Output f.

B.5 Computation of the Ate pairing

Let π_q be the Frobenius endomorphism, $\pi_q: E \to E, (x, y) \mapsto (x^q, y^q)$; let $[q]$ be the mapping: $E \to E, Q \mapsto [q]Q$; [1] unit map; the dual of π_q is π'_q, satisfying $\pi_q \cdot \pi'_q = [q]$; Ker() refers to the kernel of the mapping; let the Frobenius trace of elliptic curve $E(F_q)$ be t, and $T = t - 1$.

The computation methods for Ate pairings under various structures are given below.

B.5.1 Computation of the Ate pairing over $\mathbb{G}_2 \times \mathbb{G}_1$

Let $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - [q])$, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$. Define the Ate pairings over $\mathbb{G}_2 \times \mathbb{G}_1$ as:

$$\begin{aligned} \mathrm{Ate}\colon \mathbb{G}_2 \times \mathbb{G}_1 &\to F_{q^k}^{*}/\left(F_{q^k}^{*}\right)^{r} \\ (Q,P) &\mapsto f_{Q,T}(P)^{(q^k-1)/r}. \end{aligned}$$

The computation method for Ate pairings on $\mathbb{G}_2 \times \mathbb{G}_1$ is given below.

Input: $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - [q])$, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, an integer $T = t - 1$.

Output: $\mathrm{Ate}(Q,P)$.

a) The binary representation of T is $t_j \dots t_1 t_0$, and the most significant bit t_j is 1;;

b) Set $f = 1, V = Q$;

c) For $i = j - 1$ to 0:

  c.1) Compute $f = f^2 \cdot g_{V,V}(P), V = [2]V$;

  c.2) If $t_i = 1$, compute $f = f \cdot g_{V,Q}(Q)/g_{V+Q}(P), V = V + Q$.

d) Compute $f = f^{(q^k - 1)/r}$;

e) Output f.

B.5.2 Computation of the Ate pairing over $\mathbb{G}_1 \times \mathbb{G}_2$

For supersingular elliptic curves, the definition and technique of Ate pairings mentioned above can be directly applied; whereas for ordinary curves, $\mathbb{G}_2$ needs to be transformed to torsion curve before Ate pairings could be defined.

B5.2.1 Ate pairings on supersingular elliptic curves

Let E be a supersingular elliptic curve defined over F_q,

Let $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi'_q - [q])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi'_q - [1])$, $\mathbb{G}_T = F_{q^k}^{*}/\left(F_{q^k}^{*}\right)^{r}$, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$. Define the Ate pairings over $\mathbb{G}_1 \times \mathbb{G}_2$ as:

$$\begin{aligned} \mathrm{Ate}\colon \mathbb{G}_1 \times \mathbb{G}_2 &\to F_{q^k}^{*}/\left(F_{q^k}^{*}\right)^{r} \\ (P,Q) &\mapsto f_{P,T}(Q)^{(q^k-1)/r}. \end{aligned}$$

The computation method for Ate pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ is given below.

Input: $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi'_q - [q])$, $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi'_q - [1])$, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, an integer $T = t - 1$.

Output: $Ate(P,Q)$.

a) The binary representation of T is $t_j \dots t_1 t_0$, and the most significant bit t_j is 1;

b) Set $f = 1, V = P$;

c) For $i = j - 1$ to 0:

c.1) Compute $f = f^2 \cdot g_{V,V}(Q), V = [2]V$;

c.2) If $t_i = 1$, compute $f = f \cdot g_{V,P}(Q)/g_{V+P}(P), V = V + P$.

d) Compute $f = f^{(q^k - 1)/r}$;

e) Output f.

B.5.2.2 Ate pairings on ordinary curves

For ordinary curves, there exists an integer e, making $(\pi'_q)^e$ the automorphism on $\mathbb{G}_1$, thus, torsion curve theory could be applied to establish the relationship between $Ate(P,Q)$ and $f_{P,T^e}(Q)$, where $T = t + 1$, and t is trace.

Let E be an elliptic curve defined over F_q, E' be the d^{th} torsion curve of E, and k its embedding degree, $m = \gcd(k,d)$, $e = k/m$, ζ_m be the m^{th} primitive unit root. The value of d has three cases when $p \geq 5$:

a) $d = 6, \beta = \zeta_m^{-6}, E': y^2 = x^3 + \beta b, \phi_6: E' \to E: (x,y) \mapsto (\beta^{-1/3}x, \beta^{-1/2}y), \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1]), \mathbb{G}_2 = E'[r] \cap \mathrm{Ker}([\beta^{-1/6}]\pi_q^e - [1])$.

b) $d = 4, \beta = \zeta_m^{-4}, E': y^2 = x^3 + \beta ax, \phi_4: E' \to E: (x,y) \mapsto (\beta^{-1/2}x, \beta^{-3/4}y), \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1]), \mathbb{G}_2 = E'[r] \cap \mathrm{Ker}([\beta^{-1/4}]\pi_q^e - [1])$.

c) $d = 2, \beta = \zeta_m^{-2}, E': y^2 = x^3 + \beta^2 ax + \beta^3 b, \phi_2: E' \to E: (x,y) \mapsto (\beta^{-1}x, \beta^{-3/2}y), \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - [1]), \mathbb{G}_2 = E'[r] \cap \mathrm{Ker}([\beta^{-1/2}]\pi_q^e - [1])$.

Let $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$. The Ate pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ are defined as:

$$Ate: \mathbb{G}_1 \times \mathbb{G}_2 \to F_{q^k}^* / \left(F_{q^k}^*\right)^r$$
$$(P,Q) \mapsto f_{P,T^e}(Q)^{(q^k-1)/r}.$$

The computation method is given below.

Input: $\mathbb{G}_1$,$\mathbb{G}_2$, $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, an integer $T = t - 1$.

Output: $Ate(P,Q)$.

a) Compute $u = T^e$;

b) The binary representation of u is $t_j \ldots t_1 t_0$, and the most significant bit t_j is 1;

c) Set $f = 1, V = P$;

d) For $i = j - 1$ to 0:

d.1) Compute $f = f^2 \cdot g_{V,V}(Q), V = [2]V$;

d.2) If $t_i = 1$, compute $f = f \cdot g_{V,P}(Q)/g_{V+P}(Q), V = V + P$.

e) Compute $f = f^{(q^k-1)/r}$;

f) Output f.

If the elliptic curve on which the Ate pairings defined on $\mathbb{G}_1 \times \mathbb{G}_2$ is based is supersingular, then it is easy to see that Ate pairings are more efficient than Tate pairings. However, for ordinary curves, Ate pairings are more computationally efficient than Tate pairings only when $|T^e| \le r$, therefore, Ate pairings are recommended only when the value of t is relatively small.

B.6 Computation of the R-ate pairing

B.6.1 Definition of the R-ate pairing

The "R" in R-ate can be seen as the ratio of two pairings, and it could also be regarded as a certain fixed power of Tate pairings.

Let $A, B, a, b \in Z$, $A = aB + b$. The Miller function $f_{Q,A}(P)$ has the following features:

$$\begin{aligned} f_{Q,A}(P) &= f_{Q,aB+b}(P) = f_{Q,aB}(P) \cdot f_{Q,b}(P) \cdot g_{[aB]Q,[b]Q}(P)/g_{[A]Q}(P) \\ &= f_{Q,B}^{a}(P) \cdot f_{[B]Q,a}(P) \cdot f_{Q,b}(P) \cdot \frac{g_{[aB]Q,[b]Q}(P)}{g_{[A]Q}(P)} \end{aligned}$$

The R-ate pairing is defined as:

$$R_{A,B}(Q,P) = \left(f_{[B]Q,a}(P) \cdot f_{Q,b}(P) \cdot \frac{g_{[aB]Q,[b]Q}(P)}{g_{[A]Q}(P)}\right)^{(q^k-1)/n} = \left(\frac{f_{Q,A}(P)}{f_{Q,B}^{a}(P)}\right)^{(q^k-1)/n}.$$

If $f_{Q,A}(P)$ and $f_{Q,B}(P)$ are non-degenerate Miller functions, then $R_{A,B}(Q,P)$ is a non-degenerate pairing.

Let $L_1, L_2, M_1, M_2 \in Z$, satisfying

$$\begin{aligned} e_n^{L_1}(Q,P) &= \left(f_{Q,A}(P)\right)^{M_1\cdot(q^k-1)/n}, \\ e_n^{L_2}(Q,P) &= \left(f_{Q,B}(P)\right)^{M_2\cdot(q^k-1)/n}. \end{aligned}$$

Let $M = \mathrm{lcm}(M_1, M_2)$, $m = (M/M_1) \cdot L_1 - a(M/M_2) \cdot L_2$.

For the sake of non-degeneracy, m is not divisible by n. We have:

$$e_n^m(Q,P) = e_n^{\frac{M}{M_1}L_1 - a\frac{M}{M_2}L_2}(Q,P) = \frac{e_n(Q,P)^{L_1\frac{M}{M_1}}}{e_n(Q,P)^{aL_2\frac{M}{M_2}}} = \left(\frac{f_{Q,A}(P)}{f_{Q,B}(P)}\right)^{M\cdot(q^k-1)/n}.$$

It is easy to see that $e_n^m(Q,P) = R_{A,B}(Q,P)^M$.

Generally, a non-degenerate pairing cannot be provided by any integer pairing (A, B), and (A, B) has four cases as follows:

1. $(A, B) = (q^i, n)$
2. $(A, B) = (q, T_1)$

3. $(A, B) = (T_i, T_j)$
4. $(A, B) = (n, T_i)$.

where $T_i \equiv q^i \pmod n$, $i \in Z$, and $0 < i < k$.

Case 1: $(A, B) = \left(q^i, n\right)$, because $A = aB + b$, that is $q^i = an + b$, therefore, $b \equiv q^i \pmod n$, and

$$\left(\frac{f_{Q,q^i}(P)}{f_{Q,n}^a(P)}\right)^{(q^k-1)/n} = R_{A,B}(Q,P) = \left(f_{[n]Q,a}(P) f_{Q,b}(P) \frac{g_{[an]Q,[b]Q}(P)}{g_{[q^i]Q}(P)}\right)^{(q^k-1)/n}$$

Because $b \equiv q^i \pmod n$, $g_{[an]Q,[b]Q}(P) = g_{[q^i]Q}(P)$. Furthermore, $f_{[n]Q,a}(P) = 1$. Hence

$$R_{A,B}(Q,P) = f_{Q,q^i}(P)^{(q^k-1)/n}. \tag{1}$$

Case 2: $(A, B) = (q, T_1)$, that is $q = aT_1 + b$. Then

$$\left(\frac{f_{Q,q}(P)}{f_{Q,T_1}^a(P)}\right)^{(q^k-1)/n} = R_{A,B}(Q,P) = \left(f_{[T_1]Q,a}(P) f_{Q,b}(P) \frac{g_{[aT_1]Q,[b]Q}(P)}{g_{[q]Q}(P)}\right)^{(q^k-1)/n}.$$

Since $f_{[T_1]Q,a}(P) = f_{Q,a}^q(P)$, therefore

$$R_{A,B}(Q,P) = \left(f_{Q,a}^q(P) f_{Q,b}(P) \frac{g_{[aT_1]Q,[b]Q}(P)}{g_{[q]Q}(P)}\right)^{(q^k-1)/n}. \tag{2}$$

Case 3: $(A, B) = (T_i, T_j)$, that is $T_i = aT_j + b$, then

$$\left(\frac{f_{Q,T_i}(P)}{f_{Q,T_j}^a(P)}\right)^{(q^k-1)/n} = R_{A,B}(Q,P) = \left(f_{[T_j]Q,a}(P) f_{Q,b}(P) \frac{g_{[aT_j]Q,[b]Q}(P)}{g_{[q^i]Q}(P)}\right)^{(q^k-1)/n}.$$

Similarly, since $f_{[T_j]Q,a}(P) = f_{Q,a}^{q_j}(P)$, therefore

$$R_{A,B}(Q,P) = \left(f_{Q,a}^{q_j}(P) f_{Q,b}(P) \frac{g_{[aT_1]Q,[b]Q}(P)}{g_{[q^i]Q}(P)}\right)^{(q^k-1)/n}. \tag{3}$$

Case 4: $(A, B) = (n, T_i)$, that is $n = aT_i + b$, therefore

$$\left(\frac{f_{Q,n}(P)}{f_{Q,T_i}^a(P)}\right)^{(q^k-1)/n} = R_{A,B}(Q,P) = \left(f_{[T_i]Q,a}(P) f_{Q,b}(P) \frac{g_{[aT_i]Q,[b]Q}(P)}{g_{[n]Q}(P)}\right)^{(q^k-1)/n}.$$

Similarly, from $f_{[T_i]Q,a}(P) = f_{Q,a}^{q_i}(P)$, we have

$$R_{A,B}(Q,P) = \left(f_{Q,a}^{q_i}(P) f_{Q,b}(P) \frac{g_{[aT_i]Q,[b]Q}(P)}{g_{[n]Q}(P)}\right)^{(q^k-1)/n}. \quad (3)$$

The R-ate pairing of case 1 is also known as Ate_i pairing. Pairing computation of cases 2, 3 and 4 require two Miller loops of length $\log a$ and $\log b$ respectively. Case 2 and 4 can only alter one parameter i to obtain efficient pairings, while case 3 can alter two parameters. Therefore the R-ate pairings of case 3 are usually chosen, then $(A,B) = (T_i, T_j)$.

In order to reduce the degree of the Miller loop, various i and j can be tried to minimize the integers a and b, thus, the degree of the Miller loop could be reduced to $\log(r^{1/\Phi(k)})$.

B.6.2 Computation of the R-ate pairing on BN curves

Barreto and Naehrig put forward a method to construct ordinary curves over prime field F_q suitable for pairings, and curves constructed via this method are called BN curves. The equation of the BN curves is $E: y^2 = x^3 + b$, where $b \neq 0$. The embedding degree $k = 12$, the curve order r is a prime. The base field feature is q, the curve order is r, and the trace tr of the Frobenius mapping can be obtained by the parameter t:

$$\begin{aligned} q(t) &= 36t^4 + 24t^3 + 24t^2 + 6t + 1 \\ r(t) &= 36t^4 + 36t^3 + 18t^2 + 6t + 1 \\ tr(t) &= 6t^2 + 1 \end{aligned}$$

where $t \in Z$, such that both $q = q(t)$ and $r = r(t)$ are primes, and in order to achieve a certain security level, t must be large enough, which is at least 63 bits long.

There exists 6th order torsion curves for BN curves over F_{q^2}: $E': y^2 = x^3 + \beta b$, where $\beta \in F_{q^2}$, which is neither a square root nor cubic root in F_{q^2}, such that $r \mid \#E'(F_{q^2})$. The points in $\mathbb{G}_2$ can be represented by the points on the torsion curve E', $\phi_6: E' \to E: (x,y) \mapsto (\beta^{-1/3}x, \beta^{-1/2}y)$. Thus, the computation of pairings is restricted on the point P on $E(F_q)$ and the point Q' on $E'(F_{q^2})$.

Frobenius automorphism is π_q, and $\pi_q: E \to E, \pi_q(x,y) = (x^q, y^q), \pi_{q^2}: E \to E, \pi_{q^2}(x,y) = (x^{q^2}, y^{q^2})$.

The computation of R-ate pairing is as follows.

Input: $P \in E(F_q)[r]$, $Q \in E'(F_{q^2})[r]$, $a = 6t + 2$.

Output: $R_a(Q,P)$.

a) Suppose $a = \sum_{j=0}^{L-1} a_i 2^j, a_{L-1} = 1$;

b) Set $T = Q, f = 1$;

c) For $i = L - 2$ to 0:

c.1) Compute $f = f^2 \cdot g_{T,T}(P), T = [2]T$;

c.2) If $a_i = 1$, compute $f = f \cdot g_{T,Q}(P), T = T + Q$;

d) Compute $Q_1 = \pi_q(Q), Q_2 = \pi_{q^2}(Q)$;

e) Compute $f = f \cdot g_{T,Q_1}(P), T = T + Q_1$;

f) Compute $f = f \cdot g_{T,-Q_2}(P), T = T - Q_2$;

g) Compute $f = f^{(q^{12}-1)/r}$;

h) Output f.

For more computation methods for Weil pairings, Tate pairings, Ate pairings and R-ate pairings, please refer to (Barreto P, Lynn, Scott M. 2003), (Barreto P, Galbraith S, et al. 2004), (Eisentrager K, Lauter K, Montgomery P. 2003), (Galbraith S, Harrison K, Soldera D. 2002), (Kobayashi T, Aoki K, Imai H. 2006), (Miller V. 2004), (Scott M. 2005), (Scott M. 2006) and (Scott M, Barreto P. 2004).

B.7 Elliptic curves suitable for pairings

It is relatively easy to construct bilinear pairings for supersingular curves, yet for curves randomly generated, it is difficult to construct computable pairings. Therefore, when considering ordinary curves, ones with a structure suitable for pairings should be selected.

Assume that E is an elliptic curve defined over F_q, if the three conditions listed below are satisfied, then E is a curve suitable for pairings:

a) $\#E(F_q)$ has a prime factor r no less than $\sqrt{q}$;

b) The embedding degree of E relative to r is less than $\log_2(r)/8$;

c) The size of the largest prime factor of $r \pm 1$ equals that of r.

Below are the steps to construct elliptic curves suitable for pairings:

Step 1: Select k, compute integer t, r and q, so that there exists an elliptic curve $E(F_q)$ whose trace is t, and the curve has a subgroup of prime order r and its embedding degree is k.

Step 2: Use complex multiplication method to compute the equation parameter of this curve over F_q.

For methods to construct elliptic curves suitable for pairings, please refer to (Atkin A, Morain F. 1993), (Barreto P, Lynn B, Scott M. 2002), (Barreto P, Lynn B, Scott M. 2003), (Barreto P, Naehrig M. 2005), (Brezing F, Weng A. 2005), (Duan P, Cui S, Wah Chan C. 2005), (Dupont R, Enge A, Morain F. 2005), (Freeman D. 2006), (Freeman D, Scott M, Tesk E. 2006), (Lay G, Zimmer H. 1994), (Milne J. 2006.), (Miyaji A, Nakabayashi M, Takano S. 2001), (Scott M. 2006) and (Thuen Ø. 2006).

Annex C
(informative)

Number-theoretic algorithm

C.1 Calculation over finite fields

C.1.1 Exponentiation operation in finite fields

Let a be a positive integer, g be an element of field F_q, then the exponentiation is the process of computing g^a. By the binary method described below, exponentiation can be performed efficiently.

Input: a positive integer a, a field F_q and a field element g.

Output: g^a.

a) Set $e = a \bmod (q-1)$, if $e = 0$, then output 1;

b) The binary representation of e is $e_r e_{r-1} \dots e_1 e_0$, and the most significant bit e_r is 1;

c) Set $x = g$;

d) For $i = r-1$ to 0:

  d.1) Set $x = x^2$;

  d.2) If $e_i = 1$, set $x = g \cdot x$;

e) Output x.

For other accelerated algorithms, please refer to (Brickell et al. 1993), (Knuth 1981).

C.1.2 Inverse operation in finite fields

Let g be a nonzero element in the field F_q, then the inverse element g^{-1} is the field element c satisfying $g \cdot c = 1$. Since $c = g^{q-2}$, the inverse operation can be implemented using the exponentiation operation. Note that if q is prime and g is an integer satisfying $1 \leq g \leq q-1$, then g^{-1} is the integer c, $1 \leq c \leq q-1$, and $g \cdot c \equiv 1 \pmod{q}$.

Input: a field F_q and a nonzero field element g in F_q.

Output: the inverse element g^{-1}.

a) Compute $c = g^{q-2}$ (see C.1.1);

b) Output c.

A more efficient method is the extended Euclidean algorithm; please refer to (Knuth D. 1981).

C.1.3 Generation of Lucas sequences

Let X and Y be two nonzero integers, then the Lucas sequences U_k and V_k of X and Y are defined as follows:

$U_0 = 0, U_1 = 1$, if $k \geq 2$, $U_k = X \cdot U_{k-1} - Y \cdot U_{k-2}$;

$V_0 = 2, V_1 = X$, if $k \geq 2$, $V_k = X \cdot V_{k-1} - Y \cdot V_{k-2}$.

The recurrences above are suitable for calculating the U_k and V_k for small k's. For large integers k, the following algorithm is efficient in the calculation of U_k mod q and V_k mod q.

Input: an odd prime p, integers X and Y, a positive integer k.

Output: U_k mod q and V_k mod q.

a) Set $\Delta = X^2 - 4Y$;

b) The binary representation of k is $k_r k_{r-1} \ldots k_1 k_0$, and the most significant bit k_r is 1;

c) Set $U = 1$, $V = X$;

d) For $i = r - 1$ to 0:

d.1) Set $(U, V) = ((U \cdot V) \bmod p, (V^2 + \Delta \cdot U^2)/2) \bmod p)$;

d.2) If $k_i = 1$, set $(U, V) = (((U \cdot X + V)/2) \bmod p, (X \cdot V + \Delta \cdot U)/2) \bmod p)$;

e) Output U and V.

C.1.4 Solving square root

C.1.4.1 Solving square root on F_q

Let q be an odd prime, g be an integer satisfying $0 \leq g < q$, then the square root (mod q) of g is the integer y, where $0 \leq y < q$, such that $y^2 = g$ (mod q).

If $g = 0$, then there is only one square root, $y = 0$; if $g \neq 0$, then there are zero or two square roots (*mod* q), and if y is one root, then the other root is $q - y$.

The following algorithm can determine whether the square roots of g exist. If it exists, then the algorithm will compute one root.

Input: an odd prime q, an integer g, $0 < g < q$.

Output: if the square roots exist, output a square root mod q; otherwise output "no square root".

Algorithm 1: For $q = 3$ (mod 4), there is a positive integer u satisfying $q = 4u + 3$.

a) Compute $y = g^{u+1}$ mod q (see C.1.1);

b) Compute $z = y^2$ mod q;

c) If $z = g$, then output y; otherwise output "no square root".

Algorithm 2: For $q = 5 \pmod 8$, there is a positive integer u satisfying $q = 8u + 5$.

a) Compute $z = g^{2u+1} \bmod q$ (see C.1.1);

b) If $z = 1 \pmod q$, compute $y = g^{u+1} \bmod q$, output y and stop the algorithm;

c) If $z = -1 \pmod q$, compute $y = (2g \cdot (4g)^u) \bmod q$, output y and stop the algorithm;

d) Output "no square root".

Algorithm 3: For $q = 1 \pmod 8$, there is a positive integer u satisfying $q = 8u + 1$.

a) Set $Y = g$;

b) Generate the random value X, $0 < X < q$;

c) Compute the Lucas sequences (see B.1.3): $U = U_{4u+1} \bmod q$ and $V = V_{4u+1} \bmod q$;

d) If $V^2 = 4Y \pmod q$, then output $y = (V/2) \bmod q$ and stop the algorithm;

e) If $U \bmod q \neq 1$ and $U \bmod q \neq q - 1$, output "no square root" and stop the algorithm;

f) Go to b).

C.1.4.2 Solving square root on F_{q^2}

Let q be an odd prime, for a quadratic field extension F_{q^2}, let the reduced polynomial be $f(x) = x^2 - n, n \in F_q$, then element β of F_{q^2} can be represented as $a + bx$, $a, b \in F_q$, then the square root of β is:

$$\sqrt{\beta} = \sqrt{a+bx} = \pm\left(\sqrt{\frac{a+\sqrt{a^2-nb^2}}{2}} + \frac{xb}{2\sqrt{\frac{a+\sqrt{a^2-nb^2}}{2}}}\right), \text{ or } \pm\left(\sqrt{\frac{a-\sqrt{a^2-nb^2}}{2}} + \frac{xb}{2\sqrt{\frac{a-\sqrt{a^2-nb^2}}{2}}}\right).$$

The algorithm below can determine if β has square roots, if yes, calculate one of the roots.

Input: $\beta = a + bx \in F_{q^2}, \beta \neq 0$, an odd prime number q.

Output: if square roots of β exists, output one square root z, otherwise output "The square root does not exist".

a) Compute $U = a^2 - nb^2$;

b) Compute the square root of $U \bmod q$ (see C.1.4.1), if the square root of $U \bmod q$ exists, denoted by w_i, the equality ${w_i}^2 = U \bmod q$, $i = 1,2$ holds, go to c); otherwise output "no square root" and stop.

c) For $i = 1$ to 2:

c.1) Compute $V=(a+w_i)/2$;

c.2) Compute the square root of $U \bmod q$ (see C.1.4.1). If they exist, choose one square root y randomly, then the equality $y^2 = U \bmod q$ holds, go to d); if the square roots of $U \bmod q$ do not exist and $i = 2$, output "no square root", then stop.

d) Compute $z_1 = b/2y \pmod q$, let $z_0 = y$;

e) Output $z = z_0 + z_1 x$.

C.1.4.3 Solving square root on F_{q^m}

C.1.4.3.1 Checking square elements on F_{q^m}

Let q be an odd prime number, $m > 2$, g a nonzero element on F_{q^m}, the algorithm below can be used to check if g is a square element.

Input: an element g of the field.

Output: if g is a square element then output "square", else output "no square".

a) Compute $B = g^{(q^m-1)/2}$ (see C.1.1);

b) If $B = 1$, output "square";

c) If $B = -1$, output "no square".

C.1.4.3.2 Solving square root on F_{q^m}

Let q be an odd prime number, $m \geq 2$.

Input: an element g of the field.

Output: if g is a square element, output its square root B; otherwise, output "no square root"

a) Randomly choose a non-square element Y;

b) Compute $q^m - 1 = 2^u \times k$, k is an odd integer.

c) Compute $Y = Y^k$.

d) Compute $C = g^k$.

e) Compute $B = g^{(k+1)/2}$.

f) If $C^{2^{u-1}} \neq 1$, then output "no square root" and stop.

g) As long as $C \neq 1$:

g.1) Let I is the smallest positive integer such that $C^{2^i} = 1$;

g.2) Compute $C = C \times Y^{2^{u-i}}$;

g.3) Compute $B = B \times Y^{2^{u-i-1}}$;

h) Output B.

C.1.5 Probabilistic primality testing

Let u be a large positive integer, the following probabilistic algorithm (Miller-Rabin test) can decide whether u is a prime or a composite.

Input: a large odd u and a large positive integer T.

Output: "probably prime" or "composite".

a) Compute v and the odd w satisfying $u - 1 = 2^v \cdot w$;

b) For $j = 1$ to T:

b.1) Select a random value a in the range $[2, u - 1]$;

b.2) Set $b = a^w \bmod u$;

b.3) If $b = 1$ or $u - 1$, go to b.6);

b.4) For $i = 1$ to $v - 1$:

b.4.1) Set $b = b^2 \bmod u$;

b.4.2) If $b = u - 1$, go to b.6);

b.4.3) If $b = 1$, output "composite" and stop the algorithm;

b.4.4) The next i;

b.5) Output "composite" and stop the algorithm;

b.6) The next j;

c) Output "probably prime".

If the algorithm outputs "composite", then u is a composite. If the algorithm outputs "probably prime", then the probability of a composite u is less than 2^{-2T}. Thus, by selecting a T large enough, then the probability is negligible.

C.2 Polynomials over finite fields

C.2.1 Greatest common divisor

If $f(x) \neq 0$ and $g(x) \neq 0$ are two polynomials whose coefficients are in the field F_q, there is only one monic polynomial $d(x)$ (its coefficients are also in the field F_q) with the largest degree, and it divides $f(x)$ and $g(x)$ simultaneously. The polynomial $d(x)$ is called the greatest common divisor of $f(x)$ and $g(x)$, which is denoted by $\gcd(f(x), g(x))$. The following algorithm (the Euclidean algorithm) is used to compute the greatest common divisor of two polynomials.

Input: a finite field F_q, and two nonzero polynomials $f(x) \neq 0$ and $g(x) \neq 0$ in F_q.

Output: $d(x) = \gcd(f(x), g(x))$.

a) Set $a(x) = f(x)$, $b(x) = g(x)$;

b) When $b(t) \neq 0$, execute the loop:

b.1) Set $c(x) = a(x) \bmod b(x)$;

b.2) Set $a(x) = b(x)$;

b.3) Set $b(x) = c(x)$;

c) Let α be the coefficient of the first term in $a(x)$ and output $\alpha^{-1}a(x)$.

C.2.2 Checking irreducibility of polynomial over F_q

Let $f(x)$ be the polynomial on F_q, the following algorithm can be used to check the irreducibility of $f(x)$ efficiently.

Input: the monic polynomial $f(x)$ and a prime q.

Output: if $f(x)$ is irreducible over F_q, output "yes"; otherwise output "no".

a) Set $u(x) = x, m = \deg(f(x))$;

b) For $i = 1\ to\ \lfloor m/2 \rfloor$:

b.1) Set $u(x) = u(x)^q \bmod f(x)$;

b.2) Set $d(x) = \gcd(f(x), u(x) - x)$;

b.3) If $d(x) \neq 1$, output "no" and stop the algorithm;

c) Output "yes".

C.3 Elliptic curve algorithms

C.3.1 Finding points on elliptic curves

Given an elliptic curve over finite field, the following algorithm can be used to find a point which is not the zero point on the elliptic curve efficiently.

C.3.1.1 Finding points on $E(F_p)$.

Input: a prime p, the parameters a and b of an elliptic curve E over F_p.

Output: a nonzero point on E.

a) Select a random integer x, $0 \leq x \leq p$;

b) Set $\alpha = (x^3 + ax + b) \bmod p$;

c) If $\alpha = 0$, then output $(x, 0)$ and stop the algorithm;

d) Compute the square root of α mod p (see C.1.4.1);

e) If d) outputs "no square root", then go to a);

f) Output (x, y).

C.3.1.2 Finding points on $E(F_{q^m})$ ($m \geq 2$)

Input: finite field F_{q^m} (q is an odd prime), the parameters a and b of an elliptic curve E over F_{q^m}

Output: a nonzero point on E.

a) Select a random element x in F_{q^m}.

b) Compute $\alpha = (x^3 + ax + b)$ over F_{q^m}.

c) If $\alpha = 0$, then output $(x, 0)$ and stop the algorithm.

d) Compute the square root of α over F_{q^m}, denoted by y (see C.1.4.3);

e) If the output of d) is "no square root", then go to a);

f) Output (x, y).

C.3.2 Finding l-order points on elliptic curves

This algorithm can be used to compute the generator of l-torsion subgroup of elliptic curves.

Input: the parameters a and b of an elliptic curve E over F_q, the order of the curve $\#E(F_q) = lr$, where l is a prime number.

Output: an l-order point on $E(F_q)$.

a) Use the method of C.3.1 to select a point Q on the curve randomly.

b) Compute $P = [r]Q$;

c) If $P = O$ then go to a);

d) Output P.

C.3.3 Finding l-torsion points on twisted elliptic curves

Let $y^2 = x^3 + ax + b$ be the function of the elliptic curve E over F_{q^m}, the order $\#E(F_{q^m}) = q^m + 1 - t$. Let the equation of its twisted curve E' is $y^2 = x^3 + \beta^2 ax + \beta^3 b$, β is a non-square element of F_{q^m}, $\#E'(F_{q^m}) = q^m + 1 + t$.

Input: the parameters a, b, β of the twisted curve $E'(F_{q^m})$ of an elliptic curve $E(F_{q^m})$, the order $\#E(F_{q^m}) = n' = l \cdot r$, where l is prime.

Output: an l-order point on $E'(F_{q^m})$.

a) Use the method of C.3.1 to select a point Q on $E'(F_{q^m})$ randomly.

b) Compute $P = [r]Q$;

c) If $P = Q$ then go to a); else P is an l-torsion point.

d) Output P.

References

[1] Abdalla M, Lange T, Eds. 2012. Pairing-Based Cryptography - Pairing 2012. Proceedings (2012), vol. 7708 of Lecture Notes in Computer Science, Springer-Verlag

[2] Atkin A, Morain F. 1993. Elliptic Curves and Primality Proving, Mathematics of Computation 61(203): 29-68

[3] Barbulescu R, Gaudry P, Joux A, Thome E. 2014. A Heuristic Quasi-polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. In P. Q. Nguyen and E. Oswald, editors, Advances in Cryptology: Proceedings of EUROCRYPT '14, volume 8441 of LNCS, Springer-Verlag, 1-16

[4] Barreto P, Galbraith S, et al. 2004. Efficient Pairing Computation on Supersingular Abelian Varieties. Cryptology ePrint Archive, Report 2004/375

[5] Barreto P, Kim H, Lynn B, et al.2002. Efficient Algorithms for Pairing-based Cryptosystems, Proceedings of CRYPTO 2002, LNCS 2442. Springer-Verlag, 354-369

[6] Barreto P, Lynn B, Scott M. 2002. Constructing Elliptic Curves with Prescribed Embedding Degrees. In: Security in Communication Networks - SCN'2002, LNCS 2576. Springer-Verlag, 263-273

[7] Barreto P, Lynn B, Scott M. 2003. On the Selection of Pairing-friendly Groups. In: Selected Areas in Cryptography - SAC'2003, LNCS 3006. Ottawa, Canada: Springer-Verlag, 17-25

[8] Barreto P, Naehrig M. 2005. Pairing-friendly Elliptic Curves of Prime Order. Cryptology ePrint Archive, Report 2005/133

[9] Boneh D, Franklin M. 2001. Identity Based Encryption from the Weil-pairing, Proceedings of CRYPTO 2001, LNCS 2139. Springer-Verlag, 213-229

[10] Brezing F, Weng A. 2005. Elliptic Curves Sutable for Pairing Based Cryptography, Designs, Codes and Cryptography, 37: 133-141

[11] Brickell E, Gordon D, Mccurley K, et al. 1993. Fast Exponentiation with Precomputation. In: Advances in Cryptology - EUROCRYPT'92, LNCS 658. Berlin: Springer-Verlag, 200-207

[12] Cao Zhenfu, Zhang Fanggou, Eds. 2013. Pairing-Based Cryptography - Pairing 2013. Proceedings (2013), vol. 8365 of Lecture Notes in Computer Science, Springer-Verlag.

[13] Cha J C, Cheon J H. 2002. An Identity-based Signature from Gap Diffie-Hellman Groups, Proceedings of PKC 2002, LNCS 2567. Springer-Verlag, 18-30

[14] Cheng Qi, Wan Daqing and Zhuang Jincheng. 2014. Traps to the BGJT-Algorithm for Discrete Logarithms. ePrint 2014.

[15] Cheon, J. H. 2006. Security Analysis of the Strong Diffie-hellman Problem. In EUROCRYPT (2006), S. Vaudenay, Ed., vol. 4004 of Lecture Notes in Computer Science, Springer-Verlag, 1-11

[16] Duan P, Cui S, Wah Chan C. 2005. Special Polynomial Families for Generating More Suitable Elliptic Curves for Pairing-based Cryptosystems. Cryptology ePrint Archive, Report 2005/342

[17] Dupont R, Enge A, Morain F. 2005. Building Curves with Arbitrary Small MOV Degree over Finite Prime Fields, Journal of Cryptology, 18(2): 79-89

[18] Eisentrager K, Lauter K, Montgomery P. 2003. Fast Elliptic Curve Arithmetic and Improved Weil-pairing Evaluation. In: Topics in Cryptology, CT-RSA03, LNCS 2612. Springer-Verlag, 343-354

[19] Freeman D. 2006. Constructing Pairing-friendly Elliptic Curves with Embedding Degree 10. In: Algorithmic Number Theory Symposium - ANTS-VII, LNCS 4076. Springer-Verlag, 452-465

[20] Freeman D, Scott M, Teske E. 2006. A Taxonomy of Pairing-friendly Elliptic Curves, Cryptology ePrint Archive Report 2006/372

[21] Frey G, Müller M, Rück H. 1999. The Tate-pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems, IEEE Transactions on Information Theory, 45(5): 1717-1719

[22] Galbraith S. 2001. Supersingular Curves in Cryptography, Proceedings of Asiacrypt 2001, LNCS 2248. Springer-Verlag, 495-513

[23] Galbraith S, Harrison K, Soldera D. 2002. Implementing the Tate-pairing, Proceedings of ANTSV, LNCS 2369. Springer-Verlag, 324-337

[24] Galbraith S , Paterson K, Eds. 2008. Pairing-Based Cryptography - Pairing 2008. Proceedings (2008), vol. 5209 of Lecture Notes in Computer Science, Springer-Verlag.

[25] Googlu F, Granger R, McGuire G, and Zumbrael J. 2013. On the Function Field Sieve and the Impact of Higher Splitting Probabilities: Application to discrete logarithms in F21971. Cryptology ePrint Archive, Report 2013/074.
[26] Hess F, Smart N, Vercauteren F. 2006. The Eta-pairing Revisited. Cryptology ePrint Archive, Report 2006/110
[27] IEEE P1363: 2000 Standard for Public Key Cryptography
[28] ISO/IEC 15946-1: 2002 Information Technology—Security Techniques—Cryptographic Techniques Based on Elliptic Curves — Part 1: General
[29] ISO/IEC 15946-2: 2002 Information Technology—Security Techniques—Cryptographic Techniques Based on Elliptic Curves — Part 2: Digital Signatures
[30] ISO/IEC 15946-3: 2002 Information Technology—Security Techniques—Cryptographic Techniques Based on Elliptic Curves — Part 3: Key Establishment
[31] ISO/IEC 15946-4: 2003 Information Technology—Security Techniques—Cryptographic Techniques Based on Elliptic Curves — Part 4: Digital Signatures Giving Message Recovery
[32] ISO/IEC 14888-3: 2004 Information Technology—Security Techniques—Digital Signatures with Appendix Part 3: Discrete Logarithm Based Mechanisms
[33] ITU-T Recommendation X.680 Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation (eqv ISO/IEC 8824-1)
[34] ITU-T Recommendation X.681 Information Technology—Abstract Syntax Notation One (ASN.1): Information Object Specification (eqv ISO/IEC 8824-2)
[35] ITU-T Recommendation X.682 Information Technology—Abstract Syntax Notation One (ASN.1): Constraint Specification (eqv ISO/IEC 8824-3)
[36] ITU-T Recommendation X.683 Information Technology—Abstract Syntax Notation One (ASN.1): Parametrization of ASN.1 Specifications (eqv ISO/IEC 8824-4)
[37] ITU-T Recommendation X.690 Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (eqv ISO/IEC 8825-1)
[38] ITU-T Recommendation X.691 Information Technology—ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER) (eqv ISO/IEC 8825-2)
[39] Joux A. 2013. Faster Index Calculus for the Medium Prime Case Application to 1175-bit and 1425-bit Finite Fields. In Advances in Cryptology EUROCRYPT 2013. Springer-Verlag, 177-193
[40] Joux A. 2013. A New Index Calculus Algorithm with Complexity L(1/4 + o(1))in Very Small characteristic. In Selected Areas in Cryptography-SAC 2013, volume 8282 of Lecture Notes in Computer Science, Springer-Verlag, 355-382
[41] Joye M, Miyaji A, Otsuka A, Eds. 2010. Pairing-Based Cryptography - Pairing 2010. Proceedings (2010), vol. 6487 of Lecture Notes in Computer Science, Springer-Verlag.
[42] Knuth D. 1981. The Art of Computer Programming(Vol 2). 2nd ed. Reading(MA): Addison-Wesley
[43] Kobayashi T, Aoki K, Imai H. 2006. Efficient Algorithms for Tate-pairing. IEICE Trans. Fundamentals, E89-A
[44] Koblitz N. 1987. Elliptic Curve Cryptosystems. Mathematics of Computation, 48:203-209
[45] Lauter K, Montgomery P, Naehrig M. 2010. An Analysis of Affine Coordinates for Pairing Computation. Pairing-Based Cryptography - Pairing 2010. Proceedings (2010), vol. 6487 of Lecture Notes in Computer Science, Springer-Verlag.
[46] Lay G, Zimmer H. 1994. Constructing Elliptic Curves with Given Group Order over Large Finite Fields, In: Algorithmic Number Theory Symposium-ANTS-1, LNCS 877. Springer-Verlag, 250-263 Menezes A. 1993. Elliptic Curve Public Key Cryptosystems. Boston: Kluwer Academic Publishers
[47] Lidl R, Niederreiter H. 1983. Finite Fields. Reading(MA): Addison-Wesley Menezes A, Okamoto T, Vanstone S. 1993. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. IEEE Transactions on Information Theory, 39: 1639-1646
[48] Miller V. 2004. The Weil-pairing and its Efficient Calculation, Journal of Cryptology, 17:235-261
[49] Milne J. 2006. Complex Multiplication, http://www.jmilne.org/math
[50] Miyaji A, Nakabayashi M, Takano S. 2001. New Explicit Conditions of Elliptic Curve Traces for FR-reduction, IEICE Transactions on Fundamentals, E84-A(5): 1234-1243

[51] Müller V. 1995. Counting the Number of Points on Elliptic Curves over Finite Fields of Characteristic Greater than Three: [Doctorate Dissertation]. Saarlandes: University of Saarlandes
[52] Pollard J. 1978. Monte Carlo Methods for Index Computation mod p. Mathematics of Computation, 32: 918-924
[53] Schoof R. 1985. Elliptic Curves over Finite Fields and the Computation of Square Roots mod p.Mathematics of Computation, 44(170): 483-494
[54] Scott M. 2005. Computing the Tate-pairing. In: CT-RSA, LNCS 3376. Springer-Verlag, 293- 304
[55] Scott M. 2006, Implementing Cryptographic Pairings, ECC 2006
[56] Scott M, Barreto P. 2004. Compressed Pairings. In: Advances in Cryptology Crypto' 2004, LNCS 3152. Springer-Verlag, 140-156
[57] Scott M, Barreto P. 2006. Generating More MNT Elliptic Curves, Designs, Codes and Cryptography, 38: 209-217
[58] Shacham H, Waters B, Eds. 2009. Pairing-Based Cryptography - Pairing 2009. Proceedings (2009), vol. 5671 of Lecture Notes in Computer Science, Springer-Verlag
[59] Silverman J. 1986. The Arithmetic of Elliptic Curves. Berlin: Springer-Verlag, GTM 106
[60] Smart N. 1999. The Discrete Logarithm Problem on Elliptic Curves of Trace One. Journal of Cryptology, 12(3): 193-196
[61] Takagi T, Okamoto T, Okamoto E, and Okamoto T, Eds. 2007. Pairing-Based Cryptography - Pairing 2007. Proceedings (2007), vol. 4575 of Lecture Notes in Computer Science, Springer-Verlag
[62] Thuen Ø. 2006. Constructing Elliptic Curves over Finite Fields Using Complex Multiplication, Master of Science in Physics and Mathematics
[63] ANSI X9.63-2001 Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. American National Standards Institute
[64] ANSI X9.62-1999 Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standards Institute