GM/T 0003.1

SM2 Public Key Cryptographic Algorithms Based on Elliptic Curves

Part 1: General

Cryptography Standardization Technical Committee of China

Issued on 2012-03-21 Translated on 2024-10-30

Contents

Foreword i

1 Scope 1

2 Symbols and abbreviations 1

3 Fields and elliptic curves 2

3.1 Finite fields 2

3.2 Elliptic curves over finite fields 3

4 Data types and conversions 6

4.1 Data types 6

4.2 Data type conversions 6

5 Elliptic curve system parameters and validation 11

5.1 General requirements 11

5.2 System parameters and validation of elliptic curves over $\boldsymbol{Fp}$ 11

5.3 System parameters and validation of elliptic curves over $\boldsymbol{F2m}$ 12

6 Key pair generation and public key validation 13

6.1 Key pair generation 13

6.2 Public key validation 13

Annex A (informative) Elliptic curve basics 15

A.1 Prime field $\boldsymbol{Fp}$ 15

A.2 Binary extension field $\boldsymbol{F2m}$ 19

A.3 Elliptic curve scalar multiplication 28

A.4 Methods for solving discrete logarithm problems 32

A.5 Compression of points on elliptic curve 34

Annex B (informative) Number theoretic algorithms 36

B.1 Finite fields and modular arithmetic 36

B.2 Polynomials over finite fields 43

B.3 Elliptic curve algorithms 46

Annex C (informative) Examples of curves 48

C.1 General requirements 48

C.2 Elliptic curves over $\boldsymbol{Fp}$ 48

C.3 Elliptic curves over $\boldsymbol{F2m}$ 49

Annex D (informative) Verifiable generation of elliptic curve equation parameters and validation 51

D.1 Verifiable generation of elliptic curve equation parameters 51

D.2 Validation of elliptic curve equation parameters 52

Bibliography 54

Foreword

GM/T 0003 “SM2 public key cryptographic algorithms based on elliptic curves” consists of 5 parts:

- Part 1: General
- Part 2: Digital signature algorithm
- Part 3: Key exchange protocol
- Part 4: Public key encryption algorithm
- Part 5: Parameter definition

This section is the first part of GM/T 0003.

1 Scope

This part of GM/T 0003 specifies fundamental mathematical knowledge and cryptographic techniques involved in the SM2 public key cryptographic algorithms based on elliptic curves. The aim is to help implementing the cryptographic mechanisms specified in other parts of this standard.

This part is applicable to public key cryptographic algorithms based on elliptic curves over prime fields and binary extension fields.

2 Symbols and abbreviations

a, b: two elements of a finite field F_q, which define an elliptic curve E over F_q.
B: the MOV threshold, which is a positive integer so that solving ECDLP in F_q is at least as hard as solving DLP in F_{q^B}.
$\deg(f)$: the degree of the polynomial $f(x)$.
E: an elliptic curve over a finite field defined by a and b.
$E(F_q)$: the set composed of all rational points (including the point at infinity O) on an elliptic curve E over F_q.
ECDLP: the elliptic curve discrete logarithm problem.
F_p: the prime field with p elements.
F_q: the finite field with q elements.
F_q^*: the multiplicative group composed of all nonzero elements of F_q.
F_{2^m}: the binary extension field with 2^m elements.
G: a base point of an elliptic curve with prime order.
$\gcd(x, y)$: the greatest common divisor of x and y.
h: the cofactor which is defined as $h = \#E(F_q)/n$, where n is the order of the base point G.
$LeftRotate()$: the operation of left rotation.
$l_{\max}$: the upper bound of divisors of the cofactor h.
m: the degree of field extension of F_{2^m} over F_2.
mod $f(x)$: the operation of modulo the polynomial $f(x)$, where if $f(x)$ is a polynomial over binary fields, then all arithmetic on the coefficients should modulo 2.
mod n: the operation of modulo n, for example, $23 \bmod 7 = 2$.
n: the order of the base point G, where n is a prime factor of $\#E(F_q)$.
O: the point at infinity on an elliptic curve, which is the zero element of the elliptic curve group.
P: $P = (x_P, y_P)$ is a nonzero point on an elliptic curve whose coordinates x_P and y_P satisfy the elliptic curve equation.
$P_1 + P_2$: the addition of two points P_1 and P_2 on the elliptic curve E.
p: a prime number greater than 3.
q: number of elements in the finite field F_q.
$r_{\min}$: the lower bound of the order of G.
Tr (): the trace function.

x_P: x-coordinate of point P.

$x^{-1} \bmod n$: the unique integer y such that $x \cdot y \equiv 1 \pmod{n}$, when $1 \leq y \leq n-1$ and $\gcd(x, n) = 1$.

$x \parallel y$: the concatenation of x and y, where x and y are bit strings or byte strings.

$x \equiv y \pmod{n}$: x and y are congruent modulo n, that is $x \bmod n = y \bmod n$.

y_P: y-coordinate of point P.

$\tilde{y}_P$: compressed form of y_P.

$\mathbb{Z}_p$: residue ring of integers modulo p.

$\langle G \rangle$: cyclic group generated by G.

$[k]P$: the k multiples of the point P, that is $[k]P = \underbrace{P + P + \cdots + P}_{Add\ k\ times}$, where k is a positive integer.

$[x, y]$: set of integers which are greater than or equal to x and less than or equal to y.

$\lceil x \rceil$: ceiling function which maps x to the smallest integer greater than or equal to x. For example, $\lceil 7 \rceil = 7$ and $\lceil 8.3 \rceil = 9$.

$\lfloor x \rfloor$: floor function which maps x to the largest integer less than or equal to x. For example, $\lfloor 7 \rfloor = 7$ and $\lfloor 8.3 \rfloor = 8$.

$\#E(F_q)$: number of points on $E(F_q)$, called the order of $E(F_q)$.

$\oplus$: the bit-wise exclusive-or operator.

3 Fields and elliptic curves

3.1 Finite fields

3.1.1 Overview

This clause describes the finite field F_q and the representation of its elements, where q is an odd prime integer or a power of 2 integer. When q is an odd prime integer, it requires $q > 2^{191}$. When q is a power of 2 integer (i.e., 2^m), it requires $m > 192$ and m is a prime integer.

3.1.2 Prime field F_q

When q equals to an odd prime number p, the elements of the prime field F_p are represented by integers $0, 1, \ldots, p-1$.

a) The additive identity element is the integer 0;

b) The multiplicative identity element is the integer 1;

c) The addition operation of field elements is: $a + b = (a + b) \bmod p$, for $a, b \in F_p$;

d) The multiplication operation of field elements is: $a \cdot b = (a \cdot b) \bmod p$, for $a, b \in F_p$.

3.1.3 Binary extension field F_{2^m}

When q is 2^m, the binary extension field F_{2^m} can be seen as the m-dimensional vector space over F_2, and its elements can be represented by bit strings of length m.

The elements of F_{2^m} can be represented in many ways, and the two mostly used ways are using the polynomial basis (PB) (see Annex A.2.1.1) and the normal basis (NB) (see Annex A.2.1.3). The principle of choosing basis is to make the computation over F_{2^m} as efficient as possible. This part does not specify the choice of the basis. In the following example, the binary extension field F_{2^m} is represented by using a polynomial basis.

Suppose $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0 \quad (f_i \in F_2, i = 0,1, \ldots, m-1)$ is an irreducible polynomial over F_2, which shall be a reducible polynomial over F_{2^m}. F_{2^m} consists of all polynomials over F_2 whose degrees are less than m. The set of polynomials $\{x^{m-1}, x^{m-2}, \ldots, x, 1\}$ forms a basis for F_{2^m} over F_2, which is called the polynomial basis. For any element $a(x) = a_{m-1}x^{m-1} + \cdots + a_2x^2 + a_1x + a_0$ of F_{2^m}, its coefficients on F_2 constitute a bit string of length m, which is denoted by $a = (a_{m-1}, \ldots, a_2, a_1, a_0)$.

a) The zero element is represented by a bit string with all zeroes;

b) The multiplicative identity element is represented by a bit string $(00 \ldots 001)$;

c) The addition of two field elements is the bit-wise XOR operation of the two bit strings;

d) The multiplication of elements a and b is defined as follows: Let a and b correspond to the polynomials $a(x)$ and $b(x)$ over F_2, respectively. Then, $a \cdot b$ is defined as the bit string corresponding to the polynomial $(a(x)b(x)) \bmod f(x)$.

3.2 Elliptic curves over finite fields

The elliptic curve over a finite field is a set of points on the elliptic curve. In the affine coordinate system, a point P (which is not the point at infinity) on an elliptic curve is represented by $P = (x_P, y_P)$, where x_P and y_P are called the x-coordinate and y-coordinate of P, respectively. In this standard, F_q is called the base field.

For more details about elliptic curves, please refer to Annexes A.1 and A.2.

In this standard, all elliptic curve points are represented by affine coordinates unless otherwise specified.

3.2.1 Elliptic curves over F_p

The equation of elliptic curves over F_p (where p is a prime number greater than 3) is:

$$y^2 = x^3 + ax + b, \quad a, b \in F_p, \text{and } (4a^3 + 27b^2) \bmod p \neq 0. \qquad (1)$$

The elliptic curve $E(F_p)$ is defined as:

$$E(F_p) = \{(x, y) | x, y \in F_p \text{ ,satisfying (1)}\} \cup \{O\},$$

where O is the point at infinity.

3.2.2 Elliptic curves over F_{2^m}

The equation of an elliptic curve defined over F_{2^m} is as follows:

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in F_{2^m}, \text{and } b \neq 0. \qquad (2)$$

The elliptic curve $E(F_{2^m})$ is defined as:

$$E(F_{2^m}) = \{(x, y) | x, y \in F_{2^m}, \text{satisfying (2)}\} \cup \{O\},$$

Where O is the point at infinity. The number of points on an elliptic curve $E(F_{2^m})$ is denoted by $\#E(F_{2^m})$, which is called the order of $E(F_{2^m})$.

3.2.3 Elliptic curve group

3.2.3.1 Elliptic curve group over F_p

The points on an elliptic curve $E(F_p)$ form an abelian group under the following rules:

a) $0 + 0 = 0$;

b) $\forall P = (x, y) \in E(F_p) \backslash \{O\}, P + O = O + P = P$;

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

d) The rule for the addition of two points which are not inverse to each other: Suppose $P_1 = (x_1, y_1) \in E(F_p) \backslash \{O\}$, $P_2 = (x_2, y_2) \in E(F_p) \backslash \{O\}$, 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) The rule of doubling: Suppose $P_1 = (x_1, y_1) \in E(F_p) \backslash \{O\}$, and $y_1 \neq 0$. Let $P_3 = (x_3, y_3) = P_1 + P_1$, then

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

where

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

3.2.3.2 Elliptic curve group over F_{2^m}

The points on elliptic curve $E(\boldsymbol{F}_{2^m})$ form an Abelian group under the following rules:

a) $0 + 0 = 0$;

b) $\forall P = (x, y) \in E(\boldsymbol{F}_{2^m})\backslash\{O\}, P + O = O + P = P$;

c) $\forall P = (x, y) \in E(\boldsymbol{F}_{2^m})\backslash\{O\}$, the inverse element of P is $-P = (x, -y)$, and $P + (-P) = O$;

d) The rule for the addition of two points which are not inverse to each other: Suppose $P_1 = (x_1, y_1) \in E(\boldsymbol{F}_{2^m})\backslash\{O\}$, $P_2 = (x_2, y_2) \in E(\boldsymbol{F}_{2^m})\backslash\{O\}$, and $x_1 \neq x_2$. Let $P_3 = (x_3, y_3) = P_1 + P_2$, then

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

where

$$\lambda = \frac{y_1 + y_2}{x_1 + x_2};$$

e) The rule of doubling: Suppose $P_1 = (x_1, y_1) \in E(\boldsymbol{F}_{2^m})\backslash\{O\}$, and $x_1 \neq 0$. Let $P_3 = (x_3, y_3) = P_1 + P_1$, then

$$\begin{cases} x_3 = \lambda^2 + \lambda + a, \\ y_3 = x_1^2 + (\lambda + 1)x_3, \end{cases}$$

where

$$\lambda = x_1 + \frac{y_1}{x_1}.$$

3.2.4 Scalar multiplication on elliptic curves

The scalar multiplication on elliptic curves is the operation of adding a point to itself multiple times. Let k as a positive integer and P as a point on an elliptic curve. the scalar multiplication of P by k is to add P to itself k times, which is denoted as $Q = [k]P = \underbrace{P + P + \cdots + P}_{add\ k\ \text{times}}$. $[k]P$ can be computed recursively since $[k]P = [k-1]P + P$.

The output of scalar multiplication may be the point at infinity O.

The scalar multiplication can be implemented more efficiently. Please refer to Annex A.3 for more details.

3.2.5 Elliptic curve discrete logarithm problem

For an elliptic curve $E(F_q)$, a point $G \in E(F_q)$ of order n and a point $Q \in \langle P \rangle$, the elliptic curve discrete logarithm problem (ECDLP) is to find an integer $l \in [0, n-1]$ satisfying $Q = [l]G$.

ECDLP is closely related to the security of elliptic curve cryptosystems. Thus, it is necessary to choose secure elliptic curves. Please refer to Annex A.4 for choosing secure elliptic curves.

3.2.6 Weak elliptic curves

If there are attacking algorithms with computational complexity lower than $n^{1/2}$ (n is the order of the base point) on an elliptic curve, then the curve is called a weak elliptic curve. In accordance with this standard, weak elliptic curves are prohibited from being used.

The supersingular elliptic curves over F_q, where the characteristic of F_q divides $q + 1 - \#E(F_q)$, and the anomalous curves over F_q, where $\#E(F_q) = q$, both are weak elliptic curves.

4 Data types and conversions

4.1 Data types

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

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

Byte string: an ordered sequence of bytes, where one byte contains 8 bits.

Field element: an element of the finite field F_q.

Elliptic curve point: a pair of field elements (x_P, y_P), where x_P and y_P satisfy the elliptic 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 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 the compressed and uncompressed forms. When implemented, the hybrid form can be converted into the compressed form or uncompressed forms.

Implementation of the compressed and hybrid forms is optional in this standard. Please refer to Annex A.5 for the details of the compressed forms.

4.2 Data type conversions

Figure 1 indicates the conversion relations between the data types. The subclauses for the corresponding conversion methods are given by the marks along the arrows.

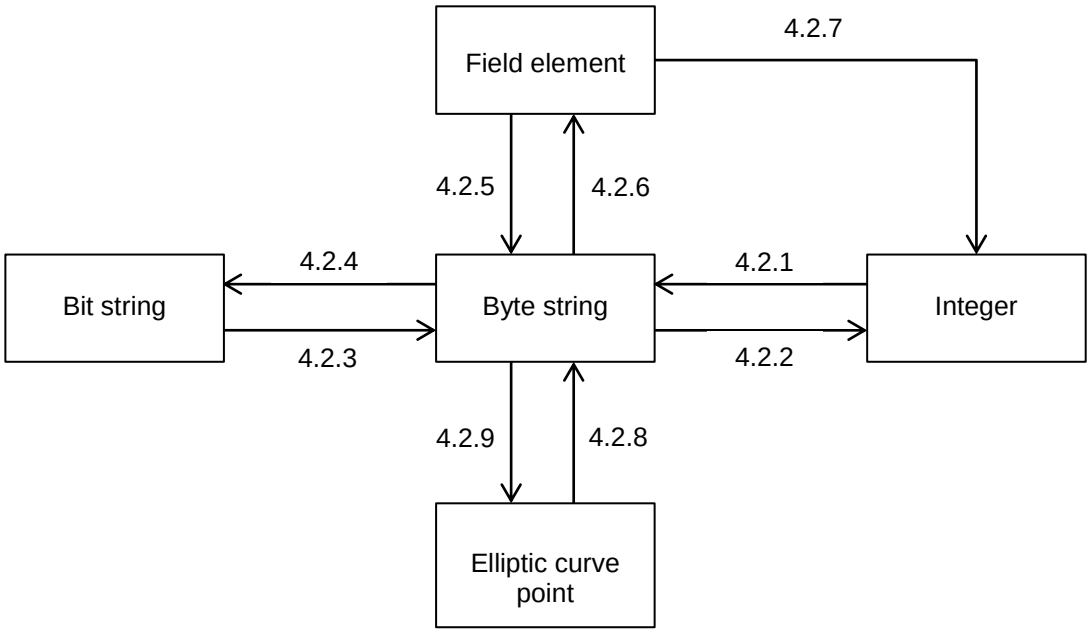

Figure 1: Data types and their conversions

4.2.1 Conversion of an integer to a byte string

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

Output: a byte string M of k bytes long.

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

b) The bytes of M satisfy

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

4.2.2 Conversion of a byte string to an integer

Input: a byte string M of k bytes long.

Output: an integer x.

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

b) Convert M to an integer x as follows:

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

4.2.3 Conversion of a bit string to a byte string

Input: a bit string s of m bits long.

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

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

b) Let $M_{k-1}, M_{k-2}, \ldots, M_0$ be the individual bytes of M from left to right. Then $M_i = s_{8i+7} s_{8i+6} \ldots s_{8i+1} s_{8i}$, where $0 \leq i < k$, and when $8i + j \geq m$ and $0 < j \leq 7$, then $s_{8i+j} = 0$.

4.2.4 Conversion of a byte string to a bit string

Input: a byte string M of k bytes long.

Output: a bit string s of m bits long, where $m = 8k$.

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

b) Let $s_{m-1}, s_{m-2}, \ldots, s_0$ 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 $j = \lfloor i/8 \rfloor$.

4.2.5 Conversion of a field element to a byte string

Input: an element α of F_q.

Output: a byte string S of $l = \lceil t/8 \rceil$ bytes long, where $t = \lceil \log_2 q \rceil$.

a) If q is an odd prime integer, then α is an integer in $[0, q-1]$. Convert α to a byte string S of l bytes long as specified in 4.2.1.

b) If $q = 2^m$, then α is a bit string of m bits long. Convert α to a byte string S of l bytes long as specified in 4.2.3.

4.2.6 Conversion of a byte string to a field element

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

Output: an element α of F_q.

a) If q is an odd prime number, convert S to an integer α as specified in 4.2.2. If $\alpha \notin [0, q-1]$, return error.

b) If $q = 2^m$, convert S to a bit string α of m bits long as specified in 4.2.4.

4.2.7 Conversion of a field element to an integer

Input: an element α of F_q.

Output: an integer x.

a) If q is an odd prime number, then $x = \alpha$. (No need to convert).

b) If $q = 2^m$, α is a bit string of m bits long. Let $s_{m-1}, s_{m-2}, \ldots, s_0$ be the individual bits of α from left to right. Then convert α to an integer x as follows:

$$x = \sum_{i=0}^{m-1} 2^i s_i.$$

4.2.8 Conversion of a point to a byte string

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

Output: a byte string S. If the uncompressed or hybrid forms are used, the length of the output byte string is $2l + 1$ bytes. If the compressed form is used, the length of the output byte string is $l + 1$ bytes, where $l = \lceil (\log_2 q)/8 \rceil$.

a) Convert the field element x_P to a byte string X_1 of l bytes long as specified in 4.2.5.

b) If the compressed form is used, then

1) Compute a bit $\tilde{y}_P$. (See Annex A.5.)

2) If $\tilde{y}_P = 0$, $PC = 02$, and if $\tilde{y}_P = 1$, $PC = 03$.

3) Output the byte string $S = PC \parallel X_1$.

c) If the uncompressed form is used, then

1) Convert the field element y_P to a byte string Y_1 of l bytes long as specified in 4.2.5.

2) Let $PC = 04$.

3) Output the byte string $S = PC \parallel X_1 \parallel Y_1$.

d) If the hybrid form is used, then

1) Convert the field element y_P to a byte string Y_1 of l bytes long as specified in 4.2.5.

2) Compute a bit $\tilde{y}_P$. (See Annex A.5.)

3) If $\tilde{y}_P = 0$, $PC = 06$, and if $\tilde{y}_P = 1$, $PC = 07$.

4) Output the byte string $S = PC \parallel X_1 \parallel Y_1$.

4.2.9 Conversion of a byte string to a point

Input: field elements a, b which define the elliptic curve over F_q and a byte string S. If the uncompressed or hybrid forms are used, the length of S is $2l + 1$ bytes. If the compressed form is used, the length of S is $l + 1$ bytes, where $l = \lceil(\log_2 q)/8\rceil$.

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

a) If the compressed form is used, $S = PC \parallel X_1$; If the uncompressed or hybrid forms are used, $S = PC \parallel X_1 \parallel Y_1$, where PC is one byte, and 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 4.2.6.

c) If the compressed form is used, then

c.1) Check if $PC = 02$ or $PC = 03$. If this is not the case, then return error.

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

c.3) Convert $(x_P, \tilde{y}_P)$ to a point (x_P, y_P) on the elliptic curve. (See Annex A.5.)

d) If the uncompressed form is used, then

d.1) Check if $PC = 04$. If this is not the case, then return error.

d.2) Convert the byte string Y_1 to a field element y_P as specified in 4.2.6.

e) If the hybrid form is used, then

e.1) Check if $PC = 06$ or $PC = 07$. If this is not the case, then return error.

e.2) Execute one of the following steps:

e.2.1) Convert the byte string Y_1 to a field element y_P as specified in 4.2.6.

e.2.2) If $PC = 06$, let $\tilde{y}_P = 0$, else if $PC = 07$, 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.5.)

f) If q is an odd prime integer, check if $y_P^2 \equiv x_P^3 + ax_P + b \pmod q$. If this is not the case, then return error. If $q = 2^m$, check if $y_P^2 + x_P y_P = x_P^3 + ax_P^2 + b$ in F_{2^m}. If this is not the case, then return error.

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

5 Elliptic curve system parameters and validation

5.1 General requirements

The elliptic curve system parameters can be public. The security of the system does not rely on the secrecy of these parameters. This standard does not specify how to generate these system parameters but specifies how to validate them. The methods of computing the order of elliptic curves and choosing the base point can be referred to Annex B.3, and the generation method of curve parameters can be referred to Annex D.

The elliptic curve system parameters can be classified into two groups as specified in the base fields:

a) Elliptic curve system parameters over F_p, if the base field is F_p (p is a prime number greater than 3);

b) Elliptic curve system parameters over F_{2^m}, if the base field is F_{2^m}.

5.2 System parameters and validation of elliptic curves over F_p

5.2.1 System parameters of elliptic curves over F_p

The system parameters of an elliptic curve over F_p include:

a) The field size $q = p$, where p is a prime number greater than 3;

b) (Optional) A bit string *SEED* of length at least 192 bits (if the elliptic curve is generated as specified in Annex D);

c) Two elements a and b belong to F_p, which define the elliptic curve equation E: $y^2 = x^3 + ax + b$;

d) The base point $G = (x_G, y_G) \in E(F_p),\ G \neq O$;

e) The base point order n satisfying $n > 2^{191}$ and $n > 4p^{1/2}$;

f) (Optional) the cofactor $h = \#E(F_p)/n$.

5.2.2 Validation of system parameters of elliptic curves over F_p

The following conditions shall be validated by the producer of the elliptic curve system parameters. The user of the elliptic curve system may selectively validate these conditions.

Input: the elliptic curve system parameters over F_p.

Output: "VALID" if the system parameters are valid; otherwise, "INVALID".

a) Verify that $q = p$ is an odd prime; (See Annex B.1.10.)

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

c) If the elliptic curve is randomly generated in a verifiable method as specified in Annex D, validate that the length of *SEED* is at least 192 bits and a, b are derived from *SEED*.

d) Verify that $4a^3 + 27b^2 \bmod p \neq 0$;

e) Verify that $y_G^2 \equiv x_G^3 + ax_G + b \ (mod\ p)$;

f) Verify that n is prime, and $n > 2^{191}$ and $n > 4p^{1/2}$; (See Annex B.1.10.)

g) Verify that $[n]G = O$; (See Annex A.3.)

h) (Optional) Compute $h' = \left\lfloor \frac{\left(p^{1/2}+1\right)^2}{n} \right\rfloor$, and validate that $h = h'$;

i) Verify that the conditions resisting the MOV attack and the anomalous curve attack are met; (See Annexes A.4.2.1 and A.4.2.2.)

j) If any validation above is failed, output "INVALID"; otherwise, output "VALID".

5.3 System parameters and validation of elliptic curves over F_{2^m}

5.3.1 System parameters of elliptic curves over F_{2^m}

The system parameters of an elliptic curve over F_{2^m} include:

a) The field size $q = 2^m$, the identifier indicating the representation of the elements in F_{2^m} (the trinomial basis (TPB), the pentanomial basis (PPB) or the Gaussian normal basis (GNB)), and an irreducible polynomial over F_2 of degree m (if TPB or PPB is used);

b) (Optional) A bit string *SEED* of length at least 192 bits (if the elliptic curve is generated as specified in Annex D);

c) Two elements a and b of F_{2^m}, which define the elliptic curve equation E: $y^2 + xy = x^3 + ax^2 + b$;

d) The base point $G = (x_G, y_G) \in E(F_{2^m}),\ G \neq O$;

e) The base point order n satisfying $n > 2^{191}$ and $n > 2^{2+m/2}$;

f) (Optional) the cofactor $h = \#E(F_{2^m})/n$.

5.3.2 Validation of system parameters of elliptic curves over F_{2^m}

The following conditions shall be validated by the producer of the elliptic curve system parameters. The user of the elliptic curve system may selectively validate these conditions.

Input: the elliptic curve system parameters over F_{2^m}.

Output: "VALID" if the system parameters are valid; otherwise, "INVALID".

a) For given m, validate that $q = 2^m$; If TPB is used, validate that the irreducible polynomial is a trinomial (see Table A.3); If PPB is used, validate that there exists no irreducible degree m trinomials and the given irreducible polynomial is pentanomial (see Table A.4); If GNB is used, validate that m is not divisible by 8;

b) Verify that a, b, x_G, y_G are bit strings of length m;

c) If the elliptic curve is randomly generated in a verifiable method as specified in Annex D, validate that the length of *SEED* is at least 192 bits and a, b both are derived from *SEED*.

d) Verify that $b \neq 0$;

e) Verify that $y_G^2 + x_G y_G \equiv x_G^3 + a x_G^2 + b$ in F_{2^m};

f) Verify that n is prime, and $n > 2^{191}$ and $n > 2^{2+m/2}$; (See Annex B.1.10.)

g) Verify that $[n]G = O$; (See Annex A.3.2.)

h) (Optional) Compute $h' = \left\lfloor \frac{\left(2^{m/2}+1\right)^2}{n} \right\rfloor$, and validate that $h = h'$;

i) Verify that the conditions resisting against the MOV attack are met; (See Annex A.4.2.1.)

j) If any validation above is failed, output "INVALID"; otherwise, output "VALID".

6 Key pair generation and public key validation

6.1 Key pair generation

Input: a set of valid elliptic curve system parameters over F_q.

Output: a key pair (d, P) related to the elliptic curve system parameters.

a) Generate an integer $d \in [1, n-2]$ using a random number generator;

b) Let G be the base point, then compute $P = (x_P, y_P) = [d]G$; (See Annex A.3.2.)

c) The key pair is (d, P), in which d is the private key and P is the public key.

6.2 Public key validation

6.2.1 Validation of public keys of elliptic curves over F_p

Input: a set of valid elliptic curve system parameters over F_p and a related public key P.

Output: "VALID" if the public key is valid; otherwise, "INVALID".

a) Verify that P is not the point at infinity O;

b) Verify that the coordinates x_P and y_P of the public key are elements belonging to F_p;

c) Verify that $y_P^2 \equiv x_P^3 + ax_P + b \pmod{p}$;

d) Verify that $[n]P = O$;

e) If all validations are passed, output "VALID"; otherwise, output "INVALID".

6.2.2 Validation of public keys of elliptic curves over F_{2^m}

Input: a set of valid elliptic curve system parameters over F_{2^m} and a related public key P.

Output: "VALID" if the public key is valid; otherwise, "INVALID".

a) Verify that P is not the point at infinity O;

b) Verify that the coordinates x_P credibility and y_P of the public key are elements belonging to F_{2^m};

c) Verify that $y_P^2 + x_P y_P \equiv x_P^3 + ax_P^2 + b$ in F_{2^m};

d) Verify that $[n]P = O$;

e) If all validations are passed, output "VALID"; otherwise, output "INVALID".

NOTE The validation of public key is optional.

Annex A

(informative)

Elliptic curve basics

A.1 Prime field F_p

A.1.1 Definition of prime field F_p

Suppose p is prime. Then F_p consists of p elements in set $\{0,1,2,\ldots,p-1\}$, which is called a prime field. 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 consisting 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. g is called 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 $F_2 = \{0,1\}$

The addition table is given in Table A.1, and the multiplication table is given in Table A.2:

Table A.1

+	0	1
0	0	1
1	1	0

Table A.2

.	0	1
0	0	0
1	0	1

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

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 Definition of elliptic curve over finite field

A.1.2.1 Overview

The elliptic curves over finite field are commonly represented in two manners, i.e., via the affine coordinate system or the projective coordinate system.

A.1.2.2 Affine coordinate

Suppose p is a prime number greater than 3 and the elliptic curve equation over F_p in the affine coordinate system has the simplified form 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 $E(F_p) = \{(x, y) \mid x, y \in F_p, y^2 = x^3 + ax + b\} \cup \{O\}$, where O is the point at infinity.

The points of $E(F_{p^m})$ form an abelian group as specified in the following addition 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 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 $P_3 = (x_3, y_3) = P_1 + P_2 \neq 0$, 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 defined 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 $P_3 = P_1 + P_2 = (x_3, y_3)$, where:

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

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

b) Let $P_1 = (10, 2)$. Then $[2]P_1 = (x_3, y_3)$, where:

$$\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{3 \times 10^2 + 1}{2 \times 2} = \frac{3 \times 5 + 1}{4} = \frac{16}{4} = 4 \pmod{19},$$
$$x_3 = 42 - 10 - 10 = -4 \equiv 15 \pmod{19},$$
$$y_3 = 4 \times (10 - 15) - 2 = -22 \equiv 16 \pmod{19}.$$

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

A.1.2.3 Projective coordinate

A.1.2.3.1 Standard projective coordinate system

Suppose p is a prime integer greater than 3 and the elliptic curve equation over F_p in the standard projective coordinate system has the simplified form as $y^2z = x^3 + axz^2 + bz^3$, where $a, b \in F_p$, satisfying $4a^3 + 27b^2 \neq 0$. The set of points on the elliptic curve is denoted by $E(F_p) = \{(x, y, z) \mid x, y, z \in F_p, y^2z = x^3 + axz^2 + bz^3\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_p$ $(u \neq 0)$ 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 coordinate can be converted to the affine coordinate as $Y^2 = X^3 + aX + b$.

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 points on $E(F_p)$ is defined as follows:

a) $O + O = O$;

b) $\forall P = (x, y, z) \in E(F_p) \backslash \{O\},\ P + O = O + P = P$;

c) $\forall P = (x, y, z) \in E(F_p) \backslash \{O\}$, the inverse of P is $-P = (ux, -uy, uz), u \in F_p\ (u \neq 0)$, and $P + (-P) = O$;

d) Let $P_1 = (x_1, y_1, z_1) \in E(F_p) \backslash \{O\},\ P_2 = (x_2, y_2, z_2) \in E(F_p) \backslash \{O\}$, and $P_3 = P_1 + P_2 = (x_3, y_3, z_3) \neq O$.

If $P_1 \neq P_2$, then

$\lambda_1 = x_1z_2,\ \lambda_2 = x_2z_1,\ \lambda_3 = \lambda_1 - \lambda_2,\ \lambda_4 = y_1z_2,\ \lambda_5 = y_2z_1,\ \lambda_6 = \lambda_4 - \lambda_5,\ \lambda_7 = \lambda_1 + \lambda_2,$
$\lambda_8 = z_1z_2,\ \lambda_9 = \lambda_3^2,\ \lambda_{10} = \lambda_3\lambda_9,\ \lambda_{11} = \lambda_8\lambda_6^2 - \lambda_7\lambda_9,\ x_3 = \lambda_3\lambda_{11},\ y_3 = \lambda_6(\lambda_9\lambda_1 - \lambda_{11}) - \lambda_4\lambda_{10},$

$z_3 = \lambda_{10}\lambda_8$.
If $P_1 = P_2$, then
$\lambda_1 = 3x_1^2 + az_1^2,\ \lambda_2 = 2y_1z_1,\ \lambda_3 = y_1^2,\ \lambda_4 = \lambda_3x_1z_1,\ \lambda_5 = \lambda_2^2,\ \lambda_6 = \lambda_1^2 - 8\lambda_4,\ x_3 = \lambda_2\lambda_6,$
$y_3 = \lambda_1(4\lambda_4 - \lambda_6) - 2\lambda_5\lambda_3,\ z_3 = \lambda_2\lambda_5$.

A.1.2.3.2 Jacobian projective coordinate system

The elliptic curve equation over F_p in the Jacobian projective coordinate system has the simplified form as $y^2 = x^3 + axz^4 + bz^6$, where $a, b \in F_p$, satisfying $4a^3 + 27b^2 \neq 0$. The set of points on the elliptic curve is denoted by $E(F_p) = \{(x, y, z) \mid x, y, z \in F_p, y^2 = x^3 + axz^4 + bz^6\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_p$ $(u \neq 0)$ such that $x_1 = u^2x_2,\ y_1 = u^3y_2$, 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 coordinate can be converted to the affine coordinate as $Y^2 = X^3 + aX + b$.

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 points on $E(F_p)$ is defined as follows:
a) $O + O = O$;
b) $\forall P = (x, y, z) \in E(F_p)\backslash\{O\},\ P + O = O + P = P$;
c) $\forall P = (x, y, z) \in E(F_p)\backslash\{O\}$, the inverse element of P is $-P = (u^2x, -u^3y, uz), u \in F_p$ $(u \neq 0)$, and $P + (-P) = O$;
d) Let $P_1 = (x_1, y_1, z_1) \in E(F_p)\backslash\{O\},\ P_2 = (x_2, y_2, z_2) \in E(F_p)\backslash\{O\}$, and $P_3 = P_1 + P_2 = (x_3, y_3, z_3) \neq O$.
If $P_1 \neq P_2$, then
$\lambda_1 = x_1z_2^2,\ \lambda_2 = x_2z_1^2,\ \lambda_3 = \lambda_1 - \lambda_2,\ \lambda_4 = y_1z_2^3,\ \lambda_5 = y_2z_1^3,\ \lambda_6 = \lambda_4 - \lambda_5,\ \lambda_7 = \lambda_1 + \lambda_2,$
$\lambda_8 = \lambda_4 + \lambda_5,\ \lambda_9 = \lambda_7\lambda_3^2,\ x_3 = \lambda_6^2 - \lambda_9,\ \lambda_{10} = \lambda_9^2 - 2x_3,\ y_3 = (\lambda_{10}\lambda_6 - \lambda_8\lambda_3^3)/2,\ z_3 = z_1z_2\lambda_3$.
If $P_1 = P_2$, then
$\lambda_1 = 3x_1^2 + az_1^4,\ \lambda_2 = 4x_1y_1^2,\ \lambda_3 = 8y_1^4,\ x_3 = \lambda_1^2 - 2\lambda_2,\ y_3 = \lambda_1(\lambda_2 - x_3) - \lambda_3,\ z_3 = 2y_1z_1$.

A.1.3 Order of elliptic curves over F_p

The order of an elliptic curve over F_p (p is a prime greater than 3) is the number of elements in the set $E(F_p)$, denoted by $\#E(F_p)$. According to Hasse's theorem, $p + 1 - 2p^{\frac{1}{2}} \leq \#E(F_p) \leq p + 1 + 2p^{1/2}$.

In the prime field F_p, if the order of a curve $\#E(F_p) = p + 1$, then this curve is called as supersingular; otherwise, it is non-supersingular.

A. 2 Binary extension field F_{2^m}

A.2.1 Definition of binary extension field F_{2^m}

The finite field F_{2^m} consisting of 2^m elements is of the m times extension of field F_2, called the degree m binary extension field. F_{2^m} can be viewed as the m-dimensional vector space over F_2. If there exist m elements $\alpha_0, \alpha_1, \dots, \alpha_{m-1}$, such that $\forall \alpha \in F_{2^m}$, α can be uniquely represented as $\alpha = a_{m-1}\alpha_{m-1} + \cdots + a_0\alpha_0 + a_1\alpha_1$ $(a_i \in F_2)$, then $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ is called a basis of F_{2^m} over F_2. There are many choices for basis. While the multiplication rules of elements in the field are different under different bases, the addition rules of elements in the field are consistent under different bases.

A.2.1.1 Polynomial basis

Suppose the irreducible polynomial $f(x)$ over F_2 is represented as $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0$ $(f_i \in F_2,\ i = 0,1, \dots, m-1)$, which is a reducible polynomial over the binary extension field F_{2^m}. The elements of F_{2^m} can be represented by all polynomials with degrees less than m, that is,

$$F_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 \mid a_i \in F_2, i = 0,1, \dots, m-1\}.$$

The set of polynomials $\{x^{m-1}, x^{m-2}, \dots, x, 1\}$ is a basis of F_{2^m} as a vector space over F_2, which is called a 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 bit string $(a_{m-1}a_{m-2} \dots a_1a_0)$ in terms of the polynomial basis. So,

$$F_{2^m} = \{(a_{m-1}a_{m-2} \dots a_1a_0) \mid a_i \in F_2, i = 0,1, \dots, m-1\}.$$

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

Addition. $\forall (a_{m-1}a_{m-2} \dots a_1a_0), (b_{m-1}b_{m-2} \dots b_1b_0) \in F_{2^m}$, then $(a_{m-1}a_{m-2} \dots a_1a_0) + (b_{m-1}b_{m-2} \dots b_1b_0) = (c_{m-1}c_{m-2} \dots c_1c_0)$ where $c_i = a_i \oplus b_i$, $i = 0,1, \dots, m-1$. That is, addition is implemented by component-wise exclusive-or.

Multiplication. $\forall (a_{m-1}a_{m-2} \dots a_1a_0), (b_{m-1}b_{m-2} \dots b_1b_0) \in F_{2^m}$, then $(a_{m-1}a_{m-2} \dots a_1a_0) \cdot (b_{m-1}b_{m-2} \dots b_1b_0) = (r_{m-1}r_{m-2} \dots r_1r_0)$, 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 $F_2[x]$.

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

Example 4: the polynomial basis representation of F_{2^5}.

Let $f(x) = x^5 + x^2 + 1$ be an irreducible polynomial over F_2. Then the elements of F_{2^5} are:

$(00000), (00001), (00010), (00011), (00100), (00101), (00110),$

$(00111), (01000), (01001), (01010), (01011), (01100), (01101),$

$(01110), (01111), (10000), (10001), (10010), (10011), (10100),$

$(10101), (10110), (10111), (11000), (11001), (11010), (11011),$

$(11100), (11101), (11110), (11111)$.

Addition: $(11011) + (10011) = (01000)$.

Multiplication: $(11011) \cdot (10011) = (00100)$

$$\begin{aligned}(x^4 + x^3 + x + 1) \cdot (x^4 + x + 1) &= x^8 + x^7 + x^4 + x^3 + x^2 + 1 \\ &= (x^5 + x^2 + 1) \cdot (x^3 + x^2 + 1) + x^2 \\ &= x^2 \pmod{f(x)}\end{aligned}$$

That is, x^2 is the reminder of $(x^4 + x^3 + x + 1) \cdot (x^4 + x + 1)$ modulo $f(x)$.

The multiplicative identity is (00001), and $\alpha = x$ is a generator of $F_{2^5}^*$. Then the powers of α are

$\alpha^0 = (00001), \alpha^1 = (00010), \alpha^2 = (00100), \alpha^3 = (01000), \alpha^4 = (10000), \alpha^5 = (00101),$

$\alpha^6 = (01010), \alpha^7 = (10100), \alpha^8 = (01101), \alpha^9 = (11010), \alpha^{10} = (10001), \alpha^{11} = (00111),$

$\alpha^{12} = (01110), \alpha^{13} = (11100), \alpha^{14} = (11101), \alpha^{15} = (11111), \alpha^{16} = (11011),$

$\alpha^{17} = (10011), \alpha^{18} = (00011), \alpha^{19} = (00110), \alpha^{20} = (01100), \alpha^{21} = (11000),$

$\alpha^{22} = (10101), \alpha^{23} = (01111), \alpha^{24} = (11110), \alpha^{25} = (11001), \alpha^{26} = (10111),$

$\alpha^{27} = (01011), \alpha^{28} = (10110), \alpha^{29} = (01001), \alpha^{30} = (10010), \alpha^{31} = (00001)$

A.2.1.2 Trinomial basis and pentanomial basis

A.2.1.2.1 Overview

The trinomial basis (TPB) and the pentanomial basis (PPB) are special polynomial bases.

A.2.1.2.2 Trinomial basis

Trinomials over F_2 are polynomials of the form $x^m + x^k + 1$, where $1 \leq k \leq m - 1$.

A trinomial basis representation of F_{2^m} is determined by an m-degree irreducible trinomial over F_2. Trinomials only exist for some specified values of m. The Example 4 above is a trinomial representation of F_{2^5}.

Table A.3 gives all the values of m, for which there exist irreducible trinomials, for $192 \leq m \leq 512$. For each such value of m, it also gives the minimum values of k such that there exist trinomials over F_2 of the form $x^m + x^k + 1$.

Table A.3

m, k	m, k	m, k	m, k	m, k	m, k
193, 15	194, 87	196, 3	198, 9	199, 34	201, 14

202, 55	204, 27	207, 43	209, 6	210, 7	212, 105
214, 73	215, 23	217, 45	218, 11	220, 7	223, 33
225, 32	228, 113	231, 26	233, 74	234, 31	236, 5
238, 73	239, 36	241, 70	242, 95	244, 111	247, 82
249, 35	250, 103	252, 15	253, 46	255, 52	257, 12
258, 71	260, 15	263, 93	265, 42	266, 47	268, 25
270, 53	271, 58	273, 23	274, 67	276, 63	278, 5
279, 5	281, 93	282, 35	284, 53	286, 69	287, 71
289, 21	292, 37	294, 33	295, 48	297, 5	300, 5
302, 41	303, 1	305, 102	308, 15	310, 93	313, 79
314, 15	316, 63	318, 45	319, 36	321, 31	322, 67
324, 51	327, 34	329, 50	330, 99	332, 89	333, 2
337, 55	340, 45	342, 125	343, 75	345, 22	346, 63
348, 103	350, 53	351, 34	353, 69	354, 99	358, 57
359, 68	362, 63	364, 9	366, 29	367, 21	369, 91
370, 139	372, 111	375, 16	377, 41	378, 43	380, 47
382, 81	383, 90	385, 6	386, 83	388, 159	390, 9
391, 28	393, 7	394, 135	396, 25	399, 26	401, 152
402, 171	404, 65	406, 141	407, 71	409, 87	412, 147
414, 13	415, 102	417, 107	418, 199	420, 7	422, 149
423, 25	425, 12	426, 63	428, 105	431, 120	433, 33
436, 165	438, 65	439, 49	441, 7	444, 81	446, 105
447, 73	449, 134	450, 47	455, 38	457, 16	458, 203
460, 19	462, 73	463, 93	465, 31	468, 27	470, 9
471, 1	473, 200	474, 191	476, 9	478, 121	479, 104
481, 138	484, 105	486, 81	487, 94	489, 83	490, 219
492, 7	494, 17	495, 76	497, 78	498, 155	500, 27
503, 3	505, 156	506, 23	508, 9	510, 69	511, 10

A.2.1.2.3 Pentanomial basis

Pentanomials over F_2 are polynomials of the form $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$, where $1 \leq k_1 < k_2 < k_3 \leq m - 1$.

A pentanomial basis representation of F_{2^m} is determined by an m-degree irreducible pentanomial over F_2. Pentanomials exist for all values m satisfying $4 \leq m \leq 512$.

Table A.4 gives all the values of m, for which there exist no irreducible trinomials, but there exist pentanomials for $192 \leq m \leq 512$. For each such value m, it also gives the values of (k_1, k_2, k_3) satisfying:

a) $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ is irreducible over F_2.

b) k_1 is as small as possible.

c) For fixed k_1, k_2 is chosen as small as possible.

d) For fixed k_1, k_2, k_3 is chosen as small as possible.

Table A.4

m (k_1,k_2,k_3)	m (k_1,k_2,k_3)	m (k_1,k_2,k_3)	m (k_1,k_2,k_3)
192 (1, 2, 7)	195 (1, 2, 37)	197 (1, 2, 21)	200 (1, 2, 81)
203 (1, 2, 45)	205 (1, 2, 21)	206 (1, 2, 63)	208 (1, 2, 83)
211 (1, 2,165)	213 (1, 2, 62)	216 (1, 2,107)	219 (1, 2, 65)
221 (1, 2, 18)	222 (1, 2, 73)	224 (1, 2,159)	226 (1, 2, 30)
227 (1, 2, 21)	229 (1, 2, 21)	230 (1, 2, 13)	232 (1, 2, 23)
235 (1, 2, 45)	237 (1, 2,104)	240 (1, 3, 49)	243 (1, 2, 17)
245 (1, 2, 37)	246 (1, 2, 11)	248 (1, 2,243)	251 (1, 2, 45)
254 (1, 2, 7)	256 (1, 2,155)	259 1, 2, 254)	261 (1, 2, 74)
262 (1, 2, 207)	264 (1, 2, 169)	267 (1, 2, 29)	269 (1, 2, 117)
272 (1, 3, 56)	275 (1, 2, 28)	277 (1, 2, 33)	280 (1, 2, 113)
283 (1, 2, 200)	285 (1, 2, 77)	288 (1, 2, 191)	290 (1, 2, 70)
291 (1, 2, 76)	293 (1, 3, 154)	296 (1, 2, 123)	298 (1, 2, 78)
299 (1, 2, 21)	301 (1, 2, 26)	304 (1, 2, 11)	306 (1, 2, 106)
307 (1, 2, 93)	309 (1, 2, 26)	311 (1, 3, 155)	312 (1, 2, 83)
315 (1, 2, 142)	317 (1, 3, 68)	320 (1, 2, 7)	323 (1, 2, 21)
325 (1, 2, 53)	326 (1, 2, 67)	328 (1, 2, 51)	331 (1, 2, 134)
334 (1, 2, 5)	335 (1, 2, 250)	336 (1, 2, 77)	338 (1, 2, 112)
339 (1, 2, 26)	341 (1, 2, 57)	344 (1, 2, 7)	347 (1, 2, 96)
349 (1, 2, 186)	352 (1, 2, 263)	355 (1, 2, 138)	356 (1, 2, 69)
357 (1, 2, 28)	360 (1, 2, 49)	361 (1, 2, 44)	363 (1, 2, 38)
365 (1, 2, 109)	368 (1, 2, 85)	371 (1, 2, 156)	373 (1, 3, 172)
374 (1, 2, 109)	376 (1, 2, 77)	379 (1, 2, 222)	381 (1, 2, 5)
384 (1, 2, 299)	387 (1, 2, 146)	389 (1, 2, 159)	392 (1, 2, 145)
395 (1, 2, 333)	397 (1, 2, 125)	398 (1, 3, 23)	400 (1, 2, 245)
403 (1, 2, 80)	405 (1, 2, 38)	408 (1, 2, 323)	410 (1, 2, 16)
411 (1, 2, 50)	413 (1, 2, 33)	416 (1, 3, 76)	419 (1, 2, 129)
421 (1, 2, 81)	424 (1, 2, 177)	427 (1, 2, 245)	429 (1, 2, 14)
430 (1, 2, 263)	432 (1, 2, 103)	434 (1, 2, 64)	435 (1, 2, 166)
437 (1, 2, 6)	440 (1, 2, 37)	442 (1, 2, 32)	443 (1, 2, 57)
445 (1, 2, 225)	448 (1, 3, 83)	451 (1, 2, 33)	452 (1, 2, 10)
453 (1, 2, 88)	454 (1, 2, 195)	456 (1, 2, 275)	459 (1, 2, 332)
461 (1, 2, 247)	464 (1, 2, 310)	466 (1, 2, 78)	467 (1, 2, 210)
469 (1, 2, 149)	472 (1, 2, 33)	475 (1, 2, 68)	477 (1, 2, 121)
480 (1, 2, 149)	482 (1, 2, 13)	483 (1, 2, 352)	485 (1, 2, 70)
488 (1, 2, 123)	491 (1, 2, 270)	493 (1, 2, 171)	496 (1, 3, 52)

499 (1, 2, 174)	501 (1, 2, 332)	502 (1, 2, 99)	504 (1, 3, 148)
507 (1, 2, 26)	509 (1, 2, 94)	512 (1, 2, 51)	

A.2.1.2.3 The rules for choosing a polynomial basis

The polynomial basis representation of F_{2^m} depends on the choice of reduced polynomials:
a) If there exist m-degree irreducible trinomials over F_2, then the reduced polynomial $f(x)$ is chosen to be a trinomial $x^m + x^k + 1$. For better implementation, k is chosen as small as possible. (These polynomials are given in Table A.3.)
b) If there doesn't exist m-degree irreducible trinomials over F_2, then the reduced polynomial $f(x)$ is chosen to be a pentanomial $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$. For better implementation, k_1, k_2, k_3 are chosen as small as possible. (These polynomials are given in Table A.4.)

A.2.1.3 Normal basis

A normal basis for F_{2^m} over F_2 is a basis of the form $\{\beta, \beta^2, \beta^{2^2} \dots, \beta^{2^{m-1}}\}$, where $\beta \in F_{2^m}$. There always exists such kind of basis. For all $\alpha \in F_{2^m}$, $\alpha = a_0\beta^{2^0} + a_1\beta^{2^1} + \cdots + a_{m-1}\beta^{2^{m-1}}$, where $a_i \in F_2, i = 0,1, \dots, m-1$. Denote $\alpha = (a_0 a_1 \dots a_{m-1})$. The field element α is represented by a bit string of length m. The field $F_{2^m} = \{(a_0 a_1 \dots a_{m-1}) | a_i \in F_2, i = 0,1, \dots, m-1\}$, the multiplicative identity is $(11 \dots 1)$, and the zero is $(00 \dots 0)$.

NOTE The order of bits in a normal basis is different from that in a polynomial basis. (See A.2.1.1.)

A.2.1.4 Gaussian normal basis

From A.2.1.3, a normal basis for F_{2^m} over F_2 is a basis of the form $\{\beta, \beta^2, \beta^{2^2} \dots, \beta^{2^{m-1}}\}$, where $\beta \in F_{2^m}$. One of the advantages of the normal basis is the efficient computation of the squaring of elements, while a basis called Gaussian normal basis is used for ordinary multiplications.

A Gaussian normal basis for F_{2^m} exists when m is not divisible by 8. Type T of a Gaussian normal basis is a positive integer measuring the complexity of multiplication. In general, multiplication is more efficient as T smaller. For given m and T, F_{2^m} has at most one Gaussian normal basis of type T. In all normal bases, there exist the most efficient algorithms of multiplication on Gaussian normal bases of type 1 and type 2. They are called optimal normal bases.

An element a of F_{2^m} is represented by a bit string $(a_{m-1} a_{m-2} \dots a_1 a_0)$ of length m under Gaussian normal bases:
a) The multiplicative identity is represented by m bits of 1.
b) The zero is represented by m bits of 0;
c) Addition is done by exclusive-or of two bit strings;
d) Multiplication is described in A.2.1.4.3.

A.2.1.4.1 The principle for choosing a normal basis

The principle for choosing a normal basis for F_{2^m} is to choose the Gaussian normal basis with the least type number if it exists. The type of Gaussian normal bases for F_{2^m}, for prime m in [192,512] are listed in Table A.5.

Table A.5

m Type	m Type	m Type	m Type	m Type	m Type
193 4	197 18	199 4	211 10	223 12	227 24
229 12	233 2	239 2	241 6	251 2	257 6
263 6	269 8	271 6	277 4	281 2	283 6
293 2	307 4	311 6	313 6	317 26	331 6
337 10	347 6	349 10	353 14	359 2	367 6
373 4	379 12	383 12	389 24	397 6	401 8
409 4	419 2	421 10	431 2	433 4	439 10
443 2	449 8	457 30	461 6	463 12	467 6
479 8	487 4	491 2	499 4	503 6	509 2

A.2.1.4.2 Gaussian normal bases test

Given the type T, the existence of Gaussian normal bases with the type T for F_{2^m} ($m > 1$ and m is not divisible by 8) can be tested using the following algorithm.

Input: m, T

Output: "YES", if there exists a Gaussian normal basis for F_{2^m} of type T; "NO", otherwise.

a) Compute $p = T \cdot m + 1$;

b) If p is not a prime number, then output "NO" and terminate.

c) Compute the order k of 2 modulo p. (See B.1.8)

d) Compute $u = T \cdot m/k$;

e) Compute $d = \gcd(u, m)$;

f) If $d = 1$, then output "YES"; otherwise, output "NO".

A.2.1.4.3 Multiplication under Gaussian normal bases

For any given Gaussian normal basis, the multiplication of two elements consists of three parts: the pre-computation, the computation of the first term c_0 of the multiplication, and the computation of the multiplication via c_0.

Pre-computation:

Input: m, T such that there exists a Gaussian normal basis B for F_{2^m} of type T.

Output: a sequence $f(1), f(2), \dots, f(p-1)$ with respect to B.

a) Compute $p = Tm + 1$;

b) Generate an integer u whose order modulo p is T. (See B.1.9.)

c) Compute a sequence $f(1), f(2), \dots, f(p-1)$:

c.1) Set $w = 1$;

c.2) For $j = 0$ to $T - 1$ do:

c.2.1) Set $n = w$;

c.2.2) For $i = 0$ to $m - 1$ do:

c.2.2.1) Set $f(n) = i$;

c.2.2.2) Set $n = 2n \bmod p$;

c.2.2.3) Set $w = uw \bmod p$;

d) Output $f(1), f(2), \dots, f(p-1)$.

Given two elements a, b represented under the Gaussian normal basis B, compute the first term c_0 of their multiplication (Denote $c_0 = F(a, b)$):

Input: m, T, a, b.

Output: the first term c_0.

a) Obtain $f(1), f(2), \dots, f(p-1)$ from pre-computation;

b) If T is even, then $J = 0$; otherwise $J = \sum_{k=1}^{m} (a_{k-1} b_{\frac{m}{2}+k-1} + a_{\frac{m}{2}+k-1} b_{k-1})$;

c) Output the formula $c_0 = J + \sum_{k=1}^{p-2} a_{f(k+1)} b_{f(p-k)}$.

Compute the multiplication of a, b via the formula of c_0:

Input: m, T, a, b.

Output: $(c_0 c_1 \dots c_{m-1}) = (a_0 a_1 \dots a_{m-1}) \times (b_0 b_1 \dots b_{m-1})$.

a) Set $(u_0 u_1 \dots u_{m-1}) = (a_0 a_1 \dots a_{m-1})$;

b) Set $(v_0 v_1 \dots v_{m-1}) = (b_0 b_1 \dots b_{m-1})$;

c) For $k = 0$ to $m - 1$ do:

c.1) Compute $c_k = F(u, v)$;

c.2) Set $u = \text{LeftRotate}(u), v = \text{LeftRotate}(v)$, where LeftRotate() is the left rotation by 1 operation;

d) Output $c = (c_0 c_1 \dots c_{m-1})$.

A.2.2 Definition of the elliptic curve over F_{2^m}

A.2.2.1 Overview

There are two common representations for the elliptic curves over F_{2^m}: the affine coordinate and the projective coordinate.

A.2.2.2 Affine coordinate

The elliptic curve equation over F_{2^m} in the affine coordinate system can be simplified as $y^2 + xy = x^3 + ax^2 + b$, where $a, b \in F_{2^m}$ and $b \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{2^m}) = \{(x, y) \mid x, y \in F_{2^m}, y^2 + xy = x^3 + ax^2 + b\} \cup \{O\}$, where O is the point at infinity.

The points on $E(F_{2^m})$ form an abelian group as specified in the following addition rules:

a) $O + O = O$;

b) $\forall P = (x, y) \in E(F_{2^m}) \backslash \{O\}$, $P + O = O + P = P$;

c) $\forall P = (x, y) \in E(F_{2^m}) \backslash \{O\}$, the inverse of P is $-P = (x, x + y), P + (-P) = O$;

d) $P_1 = (x_1, y_1) \in E(F_{2^m}) \backslash \{O\}$, $P_2 = (x_2, y_2) \in E(F_{2^m}) \backslash \{O\}$, and $x_1 \neq x_2$. Let $P_3 = (x_3, y_3) = P_1 + P_2 \neq 0$, then

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

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

e) Doubling:

Suppose $P_1 = (x_1, y_1) \in E(F_{2^m}) \backslash \{O\}$, and $x_1 \neq 0$, $P_3 = (x_3, y_3) = P_1 + P_1$, then:

$$\begin{cases} x_3 = \lambda^2 + \lambda + a, \\ y_3 = x_1^2 + (\lambda + 1)x_3, \end{cases}$$

where $\lambda = x_1 + \frac{y_1}{x_1}$.

A.2.2.3 Projective coordinate

A.2.2.3.1 Standard projective coordinate system

The elliptic curve equation over F_{2^m} in the standard projective coordinate system can be simplified as $y^2z + xyz = x^3 + ax^2z + bz^3$, where $a, b \in F_{2^m}$, and $b \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{2^m}) = \{(x, y, z) \mid x, y, z \in F_{2^m}, y^2z + xyz = x^3 + ax^2z + bz^3\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_{2^m}$ $(u \neq 0)$ 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 coordinate can be converted to the affine coordinate: $Y^2 + XY = X^3 + aX^2 + b$.

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 points on $E(F_{2^m})$ is defined as follows:

a) $O + O = O$;

b) $\forall P = (x, y, z) \in E(F_{2^m})\backslash\{O\}$, $P + O = O + P = P$;

c) $\forall P = (x, y, z) \in E(F_{2^m})\backslash\{O\}$, the inverse of P is $-P = (ux, u(x + y), uz), u \in F_{2^m}$ $(u \neq 0)$, and $P + (-P) = O$;

d) Let $P_1 = (x_1, y_1, z_1) \in E(F_{2^m})\backslash\{O\}$, $P_2 = (x_2, y_2, z_2) \in E(F_{2^m})\backslash\{O\}$, and $P_3 = P_1 + P_2 = (x_3, y_3, z_3) \neq O$.

If $P_1 \neq P_2$, then

$\lambda_1 = x_1z_2$, $\lambda_2 = x_2z_1$, $\lambda_3 = \lambda_1 + \lambda_2$, $\lambda_4 = y_1z_2$, $\lambda_5 = y_2z_1$, $\lambda_6 = \lambda_4 + \lambda_5$, $\lambda_7 = z_1z_2$, $\lambda_8 = \lambda_3^2$, $\lambda_9 = \lambda_8\lambda_7$, $\lambda_{10} = \lambda_3\lambda_8$, $\lambda_{11} = \lambda_6\lambda_7(\lambda_6 + \lambda_3) + \lambda_{10} + a\lambda_9$, $x_3 = \lambda_3\lambda_{11}$, $y_3 = \lambda_6(\lambda_1\lambda_8 + \lambda_{11}) + x_3 + \lambda_{10}\lambda_4$, $z_3 = \lambda_3\lambda_9$.

If $P_1 = P_2$, then

$\lambda_1 = x_1z_1$, $\lambda_2 = x_1^2$, $\lambda_3 = \lambda_2 + y_1z_1$, $\lambda_4 = \lambda_1^2$, $\lambda_5 = \lambda_3(\lambda_1 + \lambda_3) + a\lambda_4$, $x_3 = \lambda_1\lambda_5$, $y_3 = \lambda_2^2\lambda_1 + \lambda_3\lambda_5 + x_3$, $z_3 = \lambda_1\lambda_4$.

A.2.2.3.2 Jacobian projective coordinate system

The elliptic curve equation over F_{2^m} in the Jacobian projective coordinate system can be simplified as $y^2 + xyz = x^3 + ax^2z^2 + bz^6$, where $a, b \in F_{2^m}$, and $b \neq 0$. The set of points on the elliptic curve is denoted by $E(F_{2^m}) = \{(x, y, z) \mid x, y, z \in F_{2^m}, y^2 + xyz = x^3 + ax^2z^2 + bz^6\}$. For (x_1, y_1, z_1) and (x_2, y_2, z_2), if there is a $u \in F_{2^m}$ $(u \neq 0)$ such that $x_1 = u^2x_2$, $y_1 = u^3y_2$, 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 coordinate can be converted to the affine coordinate as $Y^2 + XY = X^3 + aX^2 + b$.

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 points on $E(F_{2^m})$ is defined as follows:

a) $O + O = O$;

b) $\forall P = (x, y, z) \in E(F_{2^m})\backslash\{O\},\ P + O = O + P = P$;

c) $\forall P = (x, y, z) \in E(F_{2^m})\backslash\{O\}$, the inverse element of P is $-P = (u^2x, u^2x + u^3y, uz), u \in F_{2^m}$ $(u \neq 0)$, and $P + (-P) = O$;

d) Let $P_1 = (x_1, y_1, z_1) \in E(F_{2^m})\backslash\{O\},\ P_2 = (x_2, y_2, z_2) \in E(F_{2^m})\backslash\{O\}$, and $P_3 = P_1 + P_2 = (x_3, y_3, z_3) \neq O$.

If $P_1 \neq P_2$, then

$\lambda_1 = x_1z_2^2,\ \lambda_2 = x_2z_1^2,\ \lambda_3 = \lambda_1 + \lambda_2,\ \lambda_4 = y_1z_2^3,\ \lambda_5 = y_2z_1^3,\ \lambda_6 = \lambda_4 + \lambda_5,\ \lambda_7 = z_1\lambda_3,\ \lambda_8 = \lambda_6x_2 + \lambda_7y_2,\ z_3 = \lambda_7z_2,\ \lambda_9 = \lambda_6 + \lambda_3,\ x_3 = az_3^3 + \lambda_6\lambda_9 + \lambda_3^3,\ y_3 = \lambda_9x_3 + \lambda_8\lambda_7^2$.

If $P_1 = P_2$, then

$z_3 = x_1z_1^2,\ x_3 = (x_1 + bz_1^2)^4,\ \lambda = z_3 + x_1^2 + y_1z_1,\ y_3 = x_1^4z_3 + \lambda x_3$.

A.2.3 Order of elliptic curves over F_{2^m}

The order of an elliptic curve over F_{2^m} is the number of elements in the set $E(F_{2^m})$, denoted by $\#E(F_{2^m})$. According to Hasse's theorem, $2^m + 1 - 2^{1+m/2} \leq \#E(F_{2^m}) \leq 2^m + 1 + 2^{1+m/2}$.

A.3 Elliptic curve scalar multiplication

A.3.1 Overview

Suppose P is a point on elliptic curve E of order N and k is a positive integer. Then P multiplied by k is Q:

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{add\ k\ times}.$$

A.3.2 Implementation of scalar multiplications on elliptic curves

There are several ways to implement the elliptic curve scalar multiplication. Three of them are given below, in which it is supposed $1 \leq k < N$.

Algorithm 1: the binary expansion method

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

Output: $Q = [k]P$.

a) Set $Q = O$;

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

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

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

c) Output Q.

Algorithm 2: the addition and subtraction method

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

Output: $Q = [k]P$.

a) Suppose the binary representation of $3k$ is $h_r h_{r-1} \dots h_1 h_0$, and the most significant bit h_r is 1.

b) The binary representation of k is $k_r k_{r-1} \dots k_1 k_0$; Obviously $r = l$ or $r = l + 1$;

c) Set $Q = P$;

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

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

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

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

e) Output Q.

NOTE Subtracting the point (x, y) is equivalent to adding the point $(x, -y)$ (in F_p) or $(x, x + y)$ (in F_{2^m}). There are several different methods to accelerate this operation.

Algorithm 3: the **sliding window method**

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

Output: $Q = [k]P$.

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 $j \geq 0$, do:

d.1) if $k_j = 0$, 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 $k_t = 1$;

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

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

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

e) Output Q.

A3.3 Estimations of the complexity of elliptic curve scalar multiplication

The complexity of point addition and doubling of elliptic curves under different coordinate systems are shown in Table A.6 and A.7, respectively.

Table A.6 Addition Complexity over Prime Fields

operation	coordinate systems		
	affine coordinate	standard projective coordinate	Jacobian projective coordinate
addition	1I+2M+1S	13M+2S	12M+4S
doubling	1I+2M+2S	8M+5S	4M+6S

Table A.7 Addition Complexity over Binary Extension Fields

operation	coordinate systems		
	affine coordinate	standard projective coordinate	Jacobian projective coordinate
addition	1I+2M+1S	15M+1S	15M+5S
doubling	1I+2M+2S	8M+3S	5M+5S

NOTE The I, M, and S in the tables stand for the inverse, multiplication, and square operations, respectively.

For the scalar multiplication $Q = [k]P$, let the bit length of k be l and the Hamming weight of k be W. Then Algorithm 1 needs $l-1$ doublings and $W-1$ additions; Algorithm 2 needs l doublings and $l/3$ additions; Algorithm 3 needs 1 doubling and $2^{r-1}-1$ additions during the pre-computation and $l-1$ doublings and $\frac{l}{r+1}-1$ additions, which is l doublings and $2^{r-1}+\frac{l}{r+1}-2$ additions in total. In general, $W \approx l/2$. The complexity of scalar multiplication is as follows (when the base field is the binary extension field and $a \neq 0$):

Algorithm 1:
When the base field is the prime field:
The complexity under affine coordinate: $1.5l$I+$3l$M+$2.5l$S
The complexity under standard projective coordinate: $14.5l$M+$6l$S
The complexity under Jacobian projective coordinate: $10l$M+$8l$S
When the base field is the binary extension field:
The complexity under affine coordinate: $1.5l$I+$3l$M+$2.5l$S
The complexity under standard projective coordinate: $15.5l$M+$3.5l$S
The complexity under Jacobian projective coordinate: $12.5l$M+$7.5l$S

Algorithm 2:
When the base field is the prime field:
The complexity under affine coordinate: $1.33l$I+$2.67l$M+$2.33l$S
The complexity under standard projective coordinate: $12.33l$M+$5.67l$S
The complexity under Jacobian projective coordinate: $8l$M+$7.33l$S
When the base field is the binary extension field:
The complexity under affine coordinate: $1.33l$I+$2.67l$M+$2.33l$S
The complexity under standard projective coordinate: $13l$M+$3.33l$S
The complexity under Jacobian projective coordinate: $10l$M+$6.67l$S

Algorithm 3:
When the base field is the prime field:
The complexity under affine coordinate: $(l+\frac{l}{r+1}+2^{r-1}-2)(2M+I+S)$+$l$S
The complexity under standard projective coordinate: $(\frac{l}{r+1}+2^{r-1}-2)(13M+2S)$+$l$(8M+5S)
The complexity under Jacobian projective coordinate: $(\frac{l}{r+1}+2^{r-1}-2)(12M+4S)$+$l$(4M+6S)
When the base field is the binary extension field:

The complexity under affine coordinate: $(l+\frac{l}{r+1}+2^{r-1}-2)(2M+I+S)+lS$

The complexity under standard projective coordinate: $(\frac{l}{r+1}+2^{r-1}-2)(15M+1S)+l$(8M+3S)

The complexity under Jacobian projective coordinate: $(\frac{l}{r+1}+2^{r-1}-2)(15M+5S)+l$(5M+5S)

A.4 Methods for solving discrete logarithm problems

A.4.1 Methods for solving elliptic curve discrete logarithm problems

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 $k \in [0, n-1]$ such that $Q = [k]P$.

The existing attacks on ECDLP are:

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

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

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

g) The 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.

For discrete logarithm problems on general curves, the current methods all have exponential complexity, and no efficient attack with sub-exponential complexity has been found; for discrete logarithm problems on some special curves, there exist algorithms with polynomial complexity or sub-exponential complexity.

When choosing the curves, the weak elliptic curves with respect to cryptography, which are vulnerable to the above attacks, shall not be used.

A.4.2 Conditions for secure elliptic curves

A.4.2.1 Condition for resisting the MOV attack

The reducing attack by A. Menezes, T. Okamoto, S. Vanstone, G. Frey and H. Ruck reduces ECDLP over F_q to DLP over F_{q^B} ($B > 1$). This attack is practical only when B is small which is not the case for most elliptic curves. The condition for resisting MOV attack is to ensure that an elliptic curve is not vulnerable to this reducing attack. Most elliptic curves over F_q satisfy this condition.

Before validating the condition, an MOV threshold should be chosen. The MOV threshold is a positive integer B such that computing DLP over F_{q^B} is at least as hard as computing ECDLP over F_q. For $q > 2^{191}$, it requires $B \geq 27$. Choosing $B \geq 27$ eliminates supersingular elliptic curves as well.

The following algorithm is used to validate that the system parameters are resistant to the MOV attack.

Input: the MOV threshold B, prime exponent q, and prime n.

Output: "CORRECT" if the elliptic curve is resistant to MOV attack; otherwise, "WRONG".

a) Set $t = 1$.

b) For i from 1 to B do:

b.1) Set $t = (t \cdot q)\ mod\ n$;

b.2) If $t = 1$, then output "WRONG" and terminate.

c) Output "CORRECT".

A.4.2.2 Condition for resisting the anomalous curve attack

Let $E(F_p)$ be an elliptic curve over the prime field $\mathrm{F_p}$. If $\#E(F_p) = p$, then $E(F_p)$ is called an anomalous curve. N. It was proved by Smart, T. Satoh and K. Araki that the DLP on the anomalous curves can be solved in polynomial time. The condition for resisting the anomalous curve attack is $\#\mathrm{E}(\mathrm{F_p}) \neq p$. Most elliptic curves over F_p satisfy this condition.

The following algorithm is used to validate that the system parameters are resistant to the anomalous curve attack.

Input: an elliptic curve $E(F_p)$ over $\mathrm{F_p}$ and its order $N = \#E(F_p)$.

Output: "CORRECT" if the elliptic curve is resistant to the anomalous curve attack; otherwise, "WRONG".

a) If $N = p$, then output "WRONG"; otherwise, output "CORRECT".

A.4.2.3 Other conditions

In order to resist the Pohlig-Hellman attack and the Pollard attack, the order of the base point n shall be a large prime; for the GHS attack, the m in F_{2^m} shall be a prime.

A.5 Compression of points on elliptic curve

A.5.1 Overview

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

A.5.2 Compression and decompression methods for points on elliptic curves over F_p

Let $P = (x_P, y_P)$ be a point on $E: y^2 = x^3 + ax + b$, 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 $\alpha = (x_P^3 + ax_P + b)$ mod p;

b) Compute the square root β of α mod p (see Annex B.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.5.3 Compression and decompression methods for points on elliptic curves F_{2^m}

Let $P = (x_P, y_P)$ be a point on $E: y^2 + xy = x^3 + ax^2 + b$ defined over F_{2^m}. If $x_P = 0$, then set $\tilde{y}_P = 0$; if $x_P \neq 0$, then set $\tilde{y}_P$ be the rightmost bit of $y_P \cdot x_P^{-1}$.

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

a) If $x_P = 0$, then $y_P = b^{2^{m-1}}$ (y_P is a square root of b in F_{2^m}.);

b) If $x_P \neq 0$, then:

b.1) Compute $\beta = x_P + a + bx_P^{-2}$ in F_{2^m}.

b.2) Find a field element z such that $z^2 + z = \beta$ (see Annex B.1.6). If no solutions exist, report an error;

b.3) Set the last bit of z as $\tilde{z}$;

b.4) If $y_P \neq \tilde{z}$, set $z = z + 1$, where 1 is the multiplicative identity.

b.5) Compute $y_P = x_P \cdot z$.

Annex B

(informative)

Number theoretic algorithms

B.1 Finite fields and modular arithmetic

B.1.1 Exponentiation operation in finite fields

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

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 $e_r e_{r-1} \ldots e_1 e_0$, and the most significant bit e_r is 1;

c) Set $x = g$;

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

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

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

e) Output x.

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

B.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 exponentiation operation. Note that if q is prime and g is an integer satisfying $1 \le g \le q-1$, then g^{-1} is the integer c, $1 \le c \le q-1$, 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 B.1.1);

b) Output c.

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

B.1.3 Generation of the Lucas sequence

Let X and Y be two nonzero integers. 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 integer k, the following algorithm can calculate $U_k \bmod q$ and $V_k \bmod q$ efficiently.

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

Output: $U_k \bmod p$ and $V_k \bmod p$.

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*, do:

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

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

e) Output U and V.

B.1.4 Solving square root of prime moduli

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

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

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

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

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

Algorithm 1: for $p \equiv 3 \pmod 4$, there is a positive integer u satisfying $p = 4u + 3$.

a) Compute $y = g^{u+1} \bmod p$ (see B.1.1);

b) Compute $z = y^2 \bmod p$;

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

Algorithm 2: for $p \equiv 5 \pmod 8$, there is a positive integer u satisfying $p = 8u + 5$.

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

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

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

d) Output "no square root".

Algorithm 3: for $p \equiv 1 \pmod 8$, there is a positive integer u satisfying $p = 8u + 1$.

a) Set $Y = g$;

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

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

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

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

f) Go to b).

B.1.5 Trace function and semi-trace function

Suppose α is an element in F_{2^m}. The trace of α is $Tr(\alpha) = \alpha + \alpha^2 + \alpha^{2^2} \ldots + \alpha^{2^{m-1}}$.

Half of the elements in F_{2^m} whose trace is 0 and another half whose trace is 1. The computation of the trace is as follows:

If the elements in F_{2^m} are represented in normal basis, then if $\alpha = (\alpha_0 \alpha_1 \cdots \alpha_{m-1})$, $Tr(\alpha) = \alpha_0 \oplus \alpha_1 \oplus \cdots \oplus \alpha_{m-1}$.

If the elements in F_{2^m} are represented in polynomial basis, then

a) Set $T = \alpha$;

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

b.1) $T = T^2 + \alpha$;

c) Output $Tr(\alpha) = T$.

If m is odd, then the semi-trace of α is $\alpha + \alpha^{2^2} + \alpha^{2^4} + \cdots + \alpha^{2^{m-1}}$.

If the elements in F_{2^m} are represented in polynomial basis, then the semi-trace can be computed via the following method.

a) Set $T = \alpha$;

b) For i from 1 to $(m-1)/2$ do:

b.1) $T = T^2$;

b.2) $T = T^2 + \alpha$;

c) Output T.

B.1.6 Solving quadratic equations over F_{2^m}

Suppose β is an element in F_{2^m}. The equation $z^2 + z = \beta$ has $2 - 2Tr(\beta)$ solutions in F_{2^m}. If $\beta = 0$, the solutions are 0 or 1; if $\beta \neq 0$, if z is one of the solutions, then $z + 1$ is a solution, too.

For given β, the following algorithm can be used to determine if the equation has solutions. If it has, then output one of them.

Input: F_{2^m} and a basis, $\beta \neq 0$.

Output: If solutions exist, output z such that $z^2 + z = \beta$; otherwise, output "no solutions".

Algorithm 1: For the normal basis representation

a) Let $\beta = (\beta_0\, \beta_1 \ldots \beta_{m-1})$;

b) Set $z_0 = 0$;

c) For i from 1 to $m-1$ do:

c.1) $z_i = z_{i-1} \oplus \beta_i$;

d) Set $z = (z_0 z_1 \ldots z_{m-1})$;

e) Compute $\gamma = z^2 + z$;

f) If $\gamma = \beta$, then output z, otherwise output "no solutions".

Algorithm 2: For polynomial basis representation (when m is odd)

a) Compute the semi-trace of $z = \beta$ (see Annex B.1.5);

b) Compute $\gamma = z^2 + z$;

c) If $\gamma = \beta$, then output z; otherwise, output "no solutions".

Algorithm 3: For any representation

a) Choose $\tau \in F_{2^m}$, such that $\tau + \tau^2 + \cdots + \tau^{2^{m-1}} = 1$;

b) Set $z = 0, w = \beta$;

c) For i from 1 to $m - 1$ do:

c.1) $z = z^2 + w^2 \cdot \tau$;

c.2) $w = w^2 + \beta$;

d) If $w \neq 0$, then output "no solution" and terminate.

e) Output z.

B.1.7 Checking the order of an integer modulo a prime

Suppose p is a prime and the integer g satisfies $1 < g < p$. The order of $g \ mod \ p$ is the least positive integer k such that $g^k \equiv 1 \ (mod \ p)$. The following algorithm is used to check that if the order of $g \ mod \ p$ is k.

Input: a prime p, a positive integer k which divides $p - 1$, and an integer $1 < g < p$.

Output: If k is the order of $g \ mod \ p$, then output "CORRECT", otherwise, output "WRONG".

a) Obtain the prime factors of k;

b) If $g^k \ mod \ p \neq 1$, the output "WRONG" and terminate;

c) For every prime factor l of k, do:

c.1) If $g^{k/l} \ mod \ p = 1$, then output "WRONG" and terminate;

d) Output "CORRECT".

B.1.8 Computing the order of an integer modulo a prime

Suppose p is a prime and the integer g satisfies $1 < g < p$. The following algorithm is used to compute the order of $g \bmod p$. The algorithm is practical when p is small.

Input: a prime p, and an integer $1 < g < p$.

Output: the order of $\bmod p$ k.

a) Set $b = g, j = 1$;

b) $b = (g \cdot b) \bmod p, j = j + 1$;

c) If $b > 1$, then go to b);

d) Output $k = j$.

B.1.9 Construction of integers with given order modulo a prime

Suppose p is a prime and T divides $p - 1$. The following algorithm can obtain an element in F_p whose order is T. The algorithm is practical when p is small.

Input: a prime p and an integer T divides $p - 1$.

Output: an integer u whose order modulo p is T.

a) Generate an integer g randomly, such that $1 < g < p$;

b) Compute the order of $g \bmod p$ k (see Annex B.1.8);

c) If k is not divisible by T, go to a);

d) Output $u = g^{k/T} \bmod p$.

B.1.10 Probabilistic primality test

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,* do:

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 $u - 1$, then go to b.6);

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

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

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

b.4.3) If $b = 1$, then output "composite" and terminate;

b.4.4) The next i;

b.5) Output "composite" and terminate;

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.

B.1.11 Approximate primality test

For a given bound l_{max}, if all the prime factors of a positive integer h are not greater than l_{max}, h is called l_{max}-smooth. For a given positive integer r_{min}, if there exists some prime $v \geq r_{min}$, such that $u = hv$, and h is l_{max}-smooth, then u is called an approximate prime. The following algorithm checks the approximate primality of u.

Input: positive integers u, l_{max}, r_{min}.

Output: If u is an approximate prime, then output h, v; otherwise, output "is not an approximate prime".

a) Set $v = u, h = 1$;

b) For l from 2 to l_{max}, do:

b.1) If l is composite, go to b.3);

b.2) If l divides v, execute the loop:

b.2.1) Set $v = v/l$ and $h = hl$;

b.2.2) If $v < r_{min}$, then output "is not an approximate prime" and terminate.

b.3) Next l.

c) If v is a probabilistic prime, then output h and v and terminate.

d) Output "is not an approximate prime".

B.2 Polynomials over finite fields

B.2.1 Greatest common divisor

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

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

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

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

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

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

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

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

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

B.2.2 Finding the roots of the irreducible polynomials over F_2 in F_{2^m}

Let $f(t)$ be a degree-m polynomial over F_2. Then $f(t)$ has m different roots in F_{2^m}. The following algorithm can be used to compute one of the roots efficiently.

Input: an irreducible polynomial $f(t)$ over F_2 and F_{2^m}.

Output: one of the roots of $f(t)$ in F_{2^m}.

a) Set $g(t) = f(t)$;

b) When $\deg(g) > 1$, execute the loop:

b.1) Choose $u \in F_{2^m}$ randomly;

b.2) Set $c(t) = ut$;

b.3) For i from 1 to $m-1$, do:

b.3.1) $c(t) = (c(t)^2 + ut) \bmod g(t)$;

b.4) Set $h(t) = \gcd(c(t), g(t))$;

b.5) If $h(t)$ is a constant or $\deg(g) = \deg(h)$, go to b.1);

b.6) If $2\deg(h) > \deg(g)$, set $g(t) = g(t)/h(t)$, otherwise set $g(t) = h(t)$;

c) Output $g(0)$.

NOTE The above operations are conducted in F_{2^m}.

B.2.3 Bases conversions

Given field F_{2^m} and two bases B_1, B_2, the following algorithm converts between B_1 and B_2.

a) Let $f(t)$ be the field polynomial in B_2, that is:

a.1) If B_2 is a polynomial basis, then let $f(t)$ be an m-degree reduced polynomial over F_2;

a.2) If B_2 is a Type-I optimal normal basis, then let $f(t) = t^m + t^{m-1} + \cdots + t + 1$.

a.3) If B_2 is a Type-II optimal normal basis, then let $f(t) = \sum_{\substack{0 \le j \le m \\ m-j \prec m+j}} t^j$, where if the binary representations of a, b are: $a = \sum u_i 2^i, b = \sum w_i 2^i$, then $a \prec b$ means $u_i \le w_i$ for all i.

a.4) If B_2 is a Gaussian normal basis of type $T \ge 3$, then:

a.4.1) Set $p = Tm + 1$;

a.4.2) Generate an integer u whose order modulo p is T. (See B.1.9.)

a.4.3) For $k = 1$ to m do:

$$e_k = \sum_{j=0}^{T-1} \exp\left(\frac{2^k u^j \pi i}{p}\right), \text{ where } i \text{ is the imaginary unit.}$$

a.4.4) Compute the polynomial $g(t) = \prod_{k=1}^{m}(t - e_k)$ (the coefficients of $g(t)$ are integers.)

a.4.5) Output $f(t) = g(t) \bmod 2$.

The complex numbers e_k should be computed with enough precision so as to be the same with every coefficient of $g(t)$. Every coefficient is an integer. It means that the biases during the computation of the coefficients should be less than $1/2$.

b) Let γ be a root of $f(t)$ with respect to B_1. (γ can be computed via the method in B.2.2.)

c) Let Γ be a matrix:

$$\Gamma = \begin{pmatrix} \gamma_{0,0} & \gamma_{0,1} & \cdots & \gamma_{0,m-1} \\ \gamma_{1,0} & \gamma_{1,1} & \cdots & \gamma_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{m-1,0} & \gamma_{m-1,1} & \cdots & \gamma_{m-1,m-1} \end{pmatrix},$$

where $\gamma_{i,j}$ are defined as follows:

c.1) If B_2 is a polynomial basis, then it can be represented as follows with respect to B_1:

$$\begin{aligned} 1 &= (\gamma_{0,0}\gamma_{0,1} \cdots \gamma_{0,m-1}) \\ \gamma &= (\gamma_{1,0}\gamma_{1,1} \cdots \gamma_{1,m-1}) \\ \gamma^2 &= (\gamma_{2,0}\gamma_{2,1} \cdots \gamma_{2,m-1}) \\ &\cdots \\ \gamma^{m-1} &= (\gamma_{m-1,0}\gamma_{m-1,1} \cdots \gamma_{m-1,m-1}) \end{aligned}$$

c.2) If B_2 is a Gaussian normal basis (with type $T \geq 1$), then it can be represented as follows with respect to B_1:

$$\begin{aligned} 1 &= (\gamma_{0,0}\gamma_{0,1} \cdots \gamma_{0,m-1}) \\ \gamma^2 &= (\gamma_{1,0}\gamma_{1,1} \cdots \gamma_{1,m-1}) \\ \gamma^4 &= (\gamma_{2,0}\gamma_{2,1} \cdots \gamma_{2,m-1}) \\ &\cdots \\ \gamma^{2^{m-1}} &= (\gamma_{m-1,0}\gamma_{m-1,1} \cdots \gamma_{m-1,m-1}) \end{aligned}$$

d) If the representation of an element with respect to B_2 is $(\beta_0\beta_1 \ldots \beta_{m-1})$, then the representation of this element with respect to B_1 is

$$(\alpha_0\alpha_1 \ldots \alpha_{m-1}) = (\beta_0\beta_1 \ldots \beta_{m-1})\Gamma.$$

If the representation of an element with respect to B_1 is $(\alpha_0\alpha_1 \ldots \alpha_{m-1})$, then the representation of this element with respect to B_2 is

$$(\beta_0\beta_1 \ldots \beta_{m-1}) = (\alpha_0\alpha_1 \ldots \alpha_{m-1})\Gamma^{-1},$$

where Γ^{-1} is the inverse of Γ modulo 2.

B.2.4 Checking irreducibility for polynomials over F_2

Let $f(x)$ be a polynomial over F_2. The following algorithm can be used to check the irreducibility of $f(x)$ efficiently.

Input: a polynomial $f(x)$ over F_2.

Output: if $f(x)$ is irreducible over F_2, then output "CORRECT"; otherwise, output "WRONG".

a) Set $d = \deg(f(x))$;

b) Set $u(x) = x$;

c) For $i = 1$ *to* $\lfloor d/2 \rfloor$, do:

c.1) Set $u(x) = u(x)^2 \bmod f(x)$;

c.2) Set $g(x) = \gcd(u(x) + x, f(x))$;

c.3) If $g(x) \neq 1$, then output "WRONG" and terminate;

d) Output "CORRECT".

B.3 Elliptic curve algorithms

B.3.1 Computing the order of elliptic curves

For random elliptic curves over finite fields, computing their orders is complicated. Currently, SEA algorithm and Satoh algorithm are two practical algorithms. For details of computing the orders, please refer to (Lehmann et al. 1994), (Muller 1995), (Satoh 2000), (Satoh 2002), (Satoh et al. 2003), (Schoof 1985) and (Schoof 1995).

B.3.2 Finding points on elliptic curves.

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

B.3.2.1 Elliptic curves over F_p

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

Output: a non-infinity 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 terminate;

d) Compute the square root of $\alpha \bmod p$ (see B.1.4.1);

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

f) Output (x, y).

B.3.2.2 Elliptic curves over F_{2^m}

Input: finite field F_{2^m}, and parameters a and b of an elliptic curve E over F_{2^m}

Output: a non-infinity point on E.

a) Select a random element x of F_{2^m}.

b) If $x = 0$, output $\left(0, b^{2^{m-1}}\right)$ and terminate;

c) Compute $\alpha = (x^3 + ax + b)$.

d) If $\alpha = 0$, then output $(x, 0)$ and terminate;

e) Set $\beta = x^{-2}\alpha$;

f) Compute z such that $z^2 + z = \beta$ (see Annex B.1.6);

g) If the output of f) is "no solutions", go to a);

h) Set $y = x \cdot z$;

i) Output (x, y).

Annex C

(informative)

Examples of curves

C.1 General requirements

In this annex, all values are represented in hexadecimal form, where the left-hand side is the most significant bit side, and the right-hand side is the least significant bit side.

C.2 Elliptic curves over F_p

Elliptic curve equation is $y^2 = x^3 + ax + b$.

Example 1: F_p-192 curve

prime p:

BDB6F4FE 3E8B1D9E 0DA8C0D4 6F4C318C EFE4AFE3 B6B8551F

a:

BB8E5E8F BC115E13 9FE6A814 FE48AAA6 F0ADA1AA 5DF91985

b:

1854BEBD C31B21B7 AEFC80AB 0ECD10D5 B1B3308E 6DBF11C1

base point $G = (x, y)$ and its order n:

x-coordinate:

4AD5F704 8DE709AD 51236DE6 5E4D4B48 2C836DC6 E4106640

y-coordinate:

02BB3A02 D4AAADAC AE24817A 4CA3A1B0 14B52704 32DB27D2

order n:

BDB6F4FE 3E8B1D9E 0DA8C0D4 0FC96219 5DFAE76F 56564677

Example 2: F_p-256 curve

prime p:

8542D69E 4C044F18 E8B92435 BF6FF7DE 45728391 5C45517D 722EDB8B 08F1DFC3

a:

787968B4 FA32C3FD 2417842E 73BBFEFF 2F3C848B 6831D7E0 EC65228B 3937E498

b:

63E4C6D3 B23B0C84 9CF84241 484BFE48 F61D59A5 B16BA06E 6E12D1DA 27C5249A

base point $G = (x, y)$ and its order n:

x-coordinate:

421DEBD6 1B62EAB6 746434EB C3CC315E 32220B3B ADD50BDC 4C4E6C14 7FEDD43D

y-coordinate:

0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2

order n:

8542D69E 4C044F18 E8B92435 BF6FF7DD 29772063 0485628D 5AE74EE7 C32E79B7

C.3 Elliptic curves over F_{2^m}

Elliptic curve equation is $y^2 + xy = x^3 + ax^2 + b$.

Example 3: F_{2^m}-193 curve

generating polynomial over the base field:

$x^{193} + x^{15} + 1$

a:

0

b:

00 2FE22037 B624DBEB C4C618E1 3FD998B1 A18E1EE0 D05C46FB

base point $G = (x, y)$ and its order n:

x-coordinate:

00 D78D47E8 5C936440 71BC1C21 2CF994E4 D21293AA D8060A84

y-coordinate:

00 615B9E98 A31B7B2F DDEEECB7 6B5D8755 86293725 F9D2FC0C

order n:

80000000 00000000 00000000 43E9885C 46BF45D8 C5EBF3A1

Example 4: F_{2^m}-257 curve

generating polynomial over the base field:

$x^{257} + x^{12} + 1$

a:

0

b:

00 E78BCD09 746C2023 78A7E72B 12BCE002 66B9627E CB0B5A25 367AD1AD 4CC6242B

base point $G = (x, y)$ and its order n:

x-coordinate:

00 CDB9CA7F 1E6B0441 F658343F 4B10297C 0EF9B649 1082400A 62E7A748 5735FADD

y-coordinate:

01 3DE74DA6 5951C4D7 6DC89220 D5F7777A 611B1C38 BAE260B1 75951DC8 060C2B3E

order n:

7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

Annex D

(informative)

Verifiable generation of elliptic curve equation parameters and validation

D.1 Verifiable generation of elliptic curve equation parameters

D.1.1 Verifiable generation of elliptic curve equation parameters over F_p

Method 1:

Input: a prime p.

Output: a bit string *SEED* and two elements a, b of F_p.

a) Choose an arbitrary bit string *SEED* of length at least 192 bits;

b) Compute $H = H_{256}(SEED)$, and denote $H = (h_{255}, h_{254}, \dots, h_0)$;

c) Set $R = \sum_{i=0}^{255} h_i 2^i$;

d) Set $r = R \bmod p$;

e) Choose two elements a, b of F_p such that $rb^2 \equiv a^3 \pmod{p}$;

f) If $(4a^3 + 27b^2) \bmod p = 0$, go to a);

g) The elliptic curve over F_p is $E: y^2 = x^3 + ax + b$;

h) Output (SEED, a, b).

Method 2:

Input: a prime p.

Output: a bit string *SEED* and two elements a, b of F_p.

a) Choose an arbitrary bit string *SEED* of length at least 192 bits;

b) Compute $H = H_{256}(SEED)$, and denote $H = (h_{255}, h_{254}, \dots, h_0)$;

c) Set $R = \sum_{i=0}^{255} h_i 2^i$;

d) Set $r = R \bmod p$;

e) Set $b = r$;

f) Set a as a fixed value of F_p;

g) If $(4a^3 + 27b^2) \bmod p = 0$, go to a);

h) The elliptic curve over F_p is $E: y^2 = x^3 + ax + b$;

i) Output (SEED, a, b).

D.1.2 Verifiable generation of elliptic curve equation parameters over F_{2^m}

Input: field size $q = 2^m$, a reduced polynomial $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0$ $(f_i \in F_2, i = 0,1, \ldots, m-1)$ of F_{2^m}.

Output: a bit string *SEED* and two elements a, b of F_{2^m}.

a) Choose an arbitrary bit string *SEED* of length at least 192;

b) Compute $H = H_{256}(SEED)$, and denote $H = (h_{255}, h_{254}, \ldots, h_0)$;

c) If $i \geq 256$, then let $h_i = 1$; set the bit string $HH = (h_{m-1}, h_{m-2}, \ldots, h_0)$ and b be the element of F_{2^m} corresponding to HH.

d) If $b = 0$, then go to a);

e) Set a as a fixed value of F_{2^m};

f) The elliptic curve over F_{2^m} is $E: y^2 + xy = x^3 + ax^2 + b$;

g) Output $(SEED, a, b)$.

D.2 Validation of elliptic curve equation parameters

D.2.1 Validation of elliptic curve equation parameters over F_p

Method 1:

Input: a bit string *SEED* and two elements a, b of F_p.

Output: "VALID" if the parameters are valid; otherwise, "INVALID".

a) Compute $H' = H_{256}(SEED)$, and denote $H' = (h_{255}, h_{254}, \ldots, h_0)$;

b) Set $R' = \sum_{i=0}^{255} h_i 2^i$;

c) Set $r' = R' \bmod p$;

d) If $r'b^2 \equiv a^3 \pmod{p}$, then output “VALID”; otherwise, output “INVALID".

Method 2:

Input: a bit string *SEED* and two elements a, b of F_p.

Output: “VALID” if the parameters are valid; otherwise, “INVALID".

a) Compute $H' = H_{256}(SEED)$, and denote $H' = (h_{255}, h_{254}, \dots, h_0)$;

b) Set $R' = \sum_{i=0}^{255} h_i 2^i$;

c) Set $r' = R' \bmod p$;

d) If $r' = b$, then output “VALID”; otherwise, output “INVALID”.

D.2.2 Validation of elliptic curve equation parameters over F_{2^m}

Input: a bit string *SEED* and two elements a, b of F_{2^m}.

Output: “VALID” if the parameters are valid; otherwise “INVALID”.

a) Compute $H' = H_{256}(SEED)$, and denote $H' = (h_{255}, h_{254}, \dots, h_0)$;

b) If $i \geq 256$, then let $h_i = 1$; set the bit string $HH' = (\mathrm{h}_{m-1}, h_{m-2}, \dots, h_0)$ and b' be the element in F_{2^m} corresponding to HH'.

c) If $b' = b$, then output “VALID”; otherwise, output “INVALID”.

NOTE In this annex, the function $H_{256}()$ is a cryptographic hash function with output size 256 bits.

Bibliography

[1] GB/T 15843.1-1999 Information technology--Security techniques--Entity authentication--Part 1:General

[2] GB/T 25069-2010 Information security techniques—Terminology

[3] Agnew G, Beth T, Mullin R,et al. 1993. Arithmetic operations in GF(2m). Journal of Cryptology,(6):3-13

[4] Agnew G, Mullin R, Onyszchuk I,et al. 1991. An implementation for a fast public-key cryptosystem.Journal of Cryptology, (3):63-79

[5] ANSI X9.62-1999 Public Key Cryptography For The Financial Services Industry：The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standards Institute

[6] ANSI X9.63-2001 Public Key Cryptography for the Financial Services Industry:Key Agreement and key Transport Using Elliptic Curve Cryptography .American Nation Standard Institute

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

[8] Blake I,Seroussi G,Smart N.1999. Elliptic Curves in Cryptography. Cambridge: Cambridge University Press

[9] ISO/IEC 15946-1: 2002 Information technology—Security techniques—Cryptographic techniques based on elliptic curves—Part 1:General

[10] ISO/IEC 15946-2: 2002 Information technology—Security techniques—Cryptographic techniques based on elliptic curves—Part 2:Digital signatures

[11] ISO/IEC 15946-3: 2002 Information technology—Security techniques—Cryptographic techniques based on elliptic curves—Part 3:Key establishment

[12] ISO/IEC 15946-4: 2003 Information technology—Security techniques—Cryptographic techniques based on elliptic curves—Part 4:Digital signatures giving message recovery

[13] ITU-T Recommendation X.680 Information Technology—Abstract Syntax Notation One (ASN.1):Specification of Basic Notation(eqv ISO/IEC 8824-1)

[14] ITU-T Recommendation X.681 Information Technology—Abstract Syntax Notation One (ASN.1):Information Object Specification(eqv ISO/IEC 8824-2)

[15] ITU-T Recommendation X.682 Information Technology—Abstract Syntax Notation One (ASN.1):Constraint Specification(eqv ISO/IEC 8824-3)

[16] ITU-T Recommendation X.683 Information Technology—Abstract Syntax Notation One (ASN.1):Parametrization of ASN.1 Specifications(eqv ISO/IEC 8824-4)

[17] 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)

[18] ITU-T Recommendation X.691 Information Technology—ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER) (eqv ISO/IEC 8825-2)

[19] Knuth D. 1981. The Art of Computer Programming .v.2. 2nd ed, Reading(MA): Addison-Wesley

[20] Koblitz N.1987. Elliptic curve cryptosystems. Mathematics of Computation, (48)203-209

[21] Lehmann F , Maurer M, Müller V, et al. 1994. Counting the number of points on elliptic curves over finite field of characteristic greater than three.In:Adleman L,Huang M D,ed. Algorithmic Number Theory. LNCS 877.Berlin: Springer-Verlag.60-70

[22] Lidl R,Niederreiter H.1987. Finite Fields. Cambridge:Cambridge University Press

[23] Mceliece R.1987. Finite Fields for Computer Scientists and Engineers. Boston:Kluwer Academic Publishers

[24] Menezes A.1993. Elliptic Curve Public Key Cryptosystems.Boston: Kluwer Academic Publishers

[25] 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

[26] 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

[27] Pollard J.1978. Monte Carlo methods for index computation mod p. Mathematics of Computation, 32:918-924

[28] Satoh T,Araki K.1998. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves.Comment.Math.Univ.St.Paul.,47(1):81-92

[29] Satoh T.2000. The canonical lift of an ordinary elliptic curve over a finite fields and its point counting. J.Ramanuijan Math.Soc.,15:247-270

[30] Satoh T. 2002. On p-adic point counting algorithms for elliptic curves over finite fields. In:Fieker C,Kohel D R,eds. Algorithmic Number Theory,LNCS 2369, Berlin: Springer-Verlag,43-66

[31] Satoh T,Skjernaa B ,Taguchi Y.2003. Fast computation of canonical lifts of elliptic curves and its application to point counting. Finite Fields Appl.,9:89~101

[32] Schoof R.1985. Elliptic curves over finite fields and the computation of square roots mod p.Mathematics of Computation, 44(170):483~494

[33] Schoof R.1995. Counting Points on Elliptic Curves over Finite Fields. Jl. de Theorie des Nombres de Bordeaux, 7:219~254

[34] Silverman J.1986. The Arithmetic of Elliptic Curves. Berlin: Springer-Verlag , GTM 106

[35] Smart N.1999. The discrete logarithm problem on elliptic curves of trace one. Journal of Cryptology,12(3):193~196

[36] ГОСТ Р 34.10-2001 ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ—КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ—Процессы формирования и проверки электронной цифровой подписи. ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССТИЙСКОЙ ФЕДЕРАЦИИ

———————————