SM2 public key cryptographic algorithm based on elliptic curves

Part 1: General

Contents

SM2 public key cryptographic algorithm based on elliptic curves

Part 1: General

1 Scope

This part of GM/T 0003 specifies fundamental mathematical knowledge and cryptographic techniques involved in the SM2 public key cryptographic algorithm 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_q: 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, $23 \mod 7 = 2$
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

$x^{-1} \mod n$: 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 $x \mod n = y \mod n$

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 $[k]P = \underbrace{P + P + \cdots + P}_{k \text{ } P\text{'s}}$, 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 maps 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 maps 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 finite field F_q and representation of its elements, where q is an odd prime number or a power of 2. When q is an odd prime number, it requires $q > 2^{191}$. When q is a power of 2 (i.e., 2^m), it requires $m > 192$ and m is a prime number.

3.1.2 Prime field F_q

When q is 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 of using polynomial basis (PB) (see Annex A.2.1.1) and normal basis (NB) (see Annex A.2.1.3). The principle of choosing basis is to make the computation in F_{2^m} as efficient as possible. This part does not specify the choice of basis. In the following example, the binary extension field F_{2^m} is represented by using 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 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 bit string with all zeroes;

b) The multiplicative identity element is represented by 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 corresponds to the polynomial $(a(x)b(x)) \bmod f(x)$.

3.2 Elliptic curves over finite fields

An elliptic curve over a finite field is composed of 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:

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

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

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

where O is the point at infinity.

3.2.2 Elliptic curves over F_{2^m}

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

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

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 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 $x_1 \neq 0$. 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 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 $y_1 \neq 0$. 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 many times. Let k be a positive integer, and P a point on an elliptic curve, P scalar multiplication by k is adding P to itself k times, which is denoted as $Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}$. $[k]P$ can be computed by recursion 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 $P \in E(F_q)$ 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 $Q = [l]P$.

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 about how to choose 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. This standard forbids the usage of weak elliptic curves.

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 $\#E(F_q) = q$, are weak elliptic curves.

4 Data types and conversions

4.1 Data types

In this standard, 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.

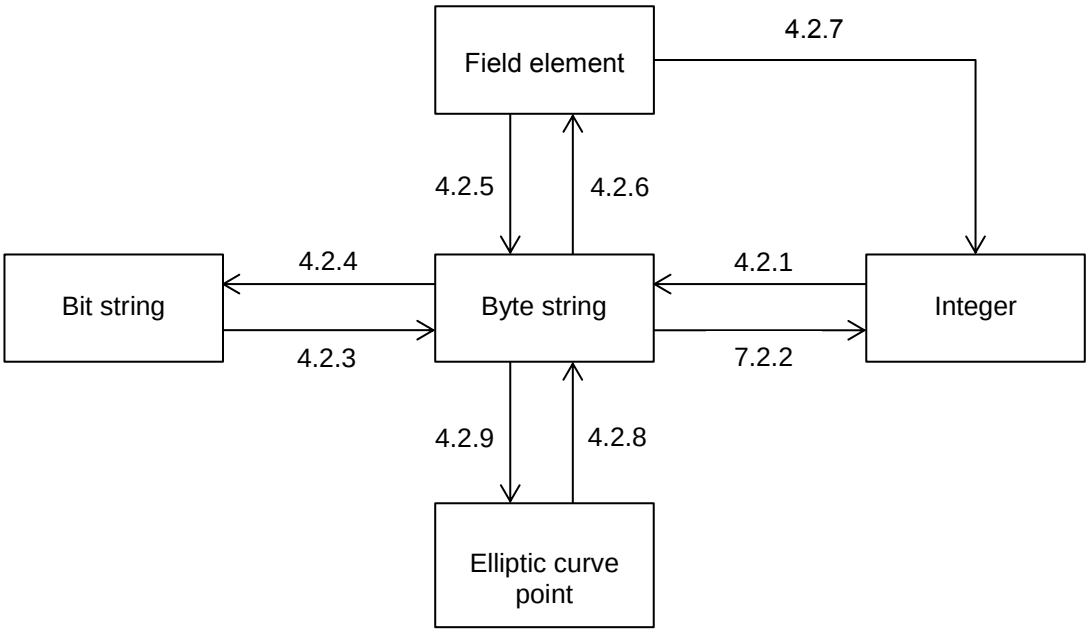
Elliptic curve point: a pair of field elements (x_P, y_P), where x_P and y_P satisfy the elliptic curve equation, or the point at infinity O.
A point can be encoded as a byte string in many forms. A byte PC is used to indicate which form is used. The byte string representation of O is a unique zero byte $PC = 00$. 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. In implementation, 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 form.

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.



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 $M_{k-1}, M_{k-2}, \ldots, M_0$ 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 $M_{k-1}, M_{k-2}, \ldots, M_0$ 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 number, 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 number, 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 $4a^3 + 27b^2 \mod p \neq 0$;

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

i) Verify that the conditions resisting against the MOV attack and the anomalous curve attack hold; (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} (trinomial basis (TPB), pentanomial basis (PPB) or 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 hold; (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 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 the 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 $r = (a + b) \mod p$, $r \in [0, p-1]$.

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

Let F_p^* be the multiplicative group consist 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}: $10, 14 \in F_{19}$, $10 + 14 = 24$, $24 \mod 19 = 5$, 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., the affine coordinate and the projective coordinate.

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 number 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

$\lambda_1 = x_1 z_2,\ \lambda_2 = x_2 z_1,\ \lambda_3 = \lambda_1 - \lambda_2,\ \lambda_4 = y_1 z_2,\ \lambda_5 = y_2 z_1,\ \lambda_6 = \lambda_4 - \lambda_5,\ \lambda_7 = \lambda_1 + \lambda_2,\ \lambda_8 = z_1 z_2,\ \lambda_9 = \lambda_3^2,\ \lambda_{10} = \lambda_3 \lambda_9,\ \lambda_{11} = \lambda_8 \lambda_6^2 - \lambda_7 \lambda_9,\ x_3 = \lambda_3 \lambda_{11},\ y_3 = \lambda_6(\lambda_9 \lambda_1 - \lambda_{11}) - \lambda_4 \lambda_{10},\ z_3 = \lambda_{10} \lambda_8$.

If $P_1 = P_2$, then

$\lambda_1 = 3x_1^2 + a z_1^2,\ \lambda_2 = 2y_1 z_1,\ \lambda_3 = y_1^2,\ \lambda_4 = \lambda_3 x_1 z_1,\ \lambda_5 = \lambda_2^2,\ \lambda_6 = \lambda_1^2 - 8\lambda_4,\ x_3 = \lambda_2 \lambda_6,\ y_3 = \lambda_1(4\lambda_4 - \lambda_6) - 2\lambda_5 \lambda_3,\ z_3 = \lambda_2 \lambda_5$.

A.1.2.3.2 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 $x_1 = u^2 x_2,\ y_1 = u^3 y_2$, and $z_1 = u z_2$, 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 $-P = (u^2 x, -u^3 y, uz), u \in F_p$ $(u \neq 0)$, 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

$\lambda_1 = x_1 z_2^2,\ \lambda_2 = x_2 z_1^2,\ \lambda_3 = \lambda_1 - \lambda_2,\ \lambda_4 = y_1 z_2^3,\ \lambda_5 = y_2 z_1^3,\ \lambda_6 = \lambda_4 - \lambda_5,\ \lambda_7 = \lambda_1 + \lambda_2,\ \lambda_8 = \lambda_4 + \lambda_5,\ \lambda_9 = \lambda_7 \lambda_3^2,\ x_3 = \lambda_6^2 - \lambda_9,\ \lambda_{10} = \lambda_9^2 - 2x_3,\ y_3 = (\lambda_{10} \lambda_6 - \lambda_8 \lambda_3^3)/2,\ z_3 = z_1 z_2 \lambda_3$.

If $P_1 = P_2$, then

$\lambda_1 = 3x_1^2 + a z_1^4,\ \lambda_2 = 4x_1 y_1^2,\ \lambda_3 = 8y_1^4,\ x_3 = \lambda_1^2 - 2\lambda_2,\ y_3 = \lambda_1(\lambda_2 - x_3) - \lambda_3,\ z_3 = 2y_1 z_1$.

A.1.3 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 by $\alpha = a_{m-1}\alpha_{m-1} + \cdots + a_0\alpha_0 + a_1\alpha_1$ $(a_i \in F_2)$, 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 degree 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: $(11011) \cdot (10011) = (00100)$

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

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

Trinomial basis (TPB) and pentanomial basis (PPB) are of 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 $1 \le k \le m - 1$.

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 $192 \le m \le 512$. 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 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 exist 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 normal basis is different from that of in 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 the 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 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 normal basis

The principle for choosing a normal basis for F_{2^m} is to choose the Gaussian normal basis with 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 type T, the existence of Gaussian normal bases with 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 $p = Tm + 1$;

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 $u = Tm/k$;

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, multiplication of two elements consists of three parts: pre-computation, computation of the first term c_0 of the multiplication, and computation of the multiplication via c_0.

Pre-computation:

Input: m, T such that there exist 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 $n = 2n \ mod \ p$;

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

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

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 $c = (c_0 c_1 \ldots c_{m-1})$.

A.2.2 Definition of 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 $-P = (u^2 x, u^2 x + u^3 y, uz), u \in F_{2^m}\ (u \neq 0)$, 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

$\lambda_1 = x_1 z_2^2,\ \lambda_2 = x_2 z_1^2,\ \lambda_3 = \lambda_1 + \lambda_2,\ \lambda_4 = y_1 z_2^3,\ \lambda_5 = y_2 z_1^3,\ \lambda_6 = \lambda_4 + \lambda_5,\ \lambda_7 = z_1 \lambda_3,\ \lambda_8 = \lambda_6 x_2 + \lambda_7 y_2,\ z_3 = \lambda_7 z_2,\ \lambda_9 = \lambda_6 + \lambda_3,\ x_3 = a z_3^3 + \lambda_6 \lambda_9 + \lambda_3^3,\ y_3 = \lambda_9 x_3 + \lambda_8 \lambda_7^2$.

If $P_1 = P_2$, then

$z_3 = x_1 z_1^2,\ x_3 = (x_1 + b z_1^2)^4,\ \lambda = z_3 + x_1^2 + y_1 z_1,\ y_3 = x_1^4 z_3 + \lambda x_3$.

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:

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k\ P's}.$$

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

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

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

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

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

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

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

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

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

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

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

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

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 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 resisting the Pohlig-Hellman attack and the Pollard attack, the order of the base point n shall be a large prime; and 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 $\alpha \mod p$ (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 $e = a \mod (q-1)$, 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 $U_k \mod q$ and $V_k \mod q$ efficiently.

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

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

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 $(U, V) = ((U \cdot V) \mod p,\ (V^2 + \Delta \cdot U^2)/2) \mod p)$;

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

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 $p = 3 \pmod 4$, 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 $p = 5 \pmod 8$, there is a positive integer u satisfying $p = 8u + 5$.

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

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

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

d) Output "no square root".

Algorithm 3: for $p = 1 \pmod 8$, 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 $V^2 = 4Y \pmod p$, 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 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 have 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 $\beta = (\beta_0 \ \beta_1 \dots \beta_{m-1})$;

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 $z = (z_0 z_1 \dots z_{m-1})$;

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 solutions" 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 $g \ mod \ p$. The algorithm is practical when p is small.

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

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

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

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

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 $g \ mod \ p \ k$ (see Annex B.1.8);

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

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

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 $b = a^w \mod u$;

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 $b = b^2 \mod u$;

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

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

b.4.4) The next i;

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

b.6) The next j;

c) Output "probably prime".

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

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 $c(t) = a(t) \mod b(t)$;

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) $c(t) = (c(t)^2 + ut) \ mod \ g(t)$;

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 $f(t) = g(t) \ mod \ 2$.

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 $m = \deg(f(x))$;

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

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

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

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 are 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 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 nonzero point on E.

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

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

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

d) Compute the square root of $\alpha \mod p$ (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 nonzero 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 appendix, 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 1: 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 1: F_{2^m}-193 curve

generating polynomial over 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 1: F_{2^m}-257 curve

generating polynomial over 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 $H = H_{256}(\text{SEED})$, and denote $H = (h_{255}, h_{254}, \dots, h_0)$;

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

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

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

f) If $(4a^3 + 27b^2) \mod p = 0$, 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 $H = H_{256}(\text{SEED})$, and denote $H = (h_{255}, h_{254}, \dots, h_0)$;

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

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

e) Set $b = r$;

f) Set a as a fixed value of F_p;

g) If $(4a^3 + 27b^2) \mod p = 0$, 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 $H = H_{256}(\text{SEED})$, 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 $H' = H_{256}(\text{SEED})$, and denote $H' = (h_{255}, h_{254}, \ldots, h_0)$;

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

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

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

b) If $i \geq 256$, then let $h_i = 1$; set the bit string $HH' = (\text{h}_{m-1}, h_{m-2}, \dots, h_0)$ 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 信息技术 安全技术 实体鉴别 第 1 部分：概述

[2] GB/T 25069-2010 信息安全技术 术语

[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

Public key cryptographic algorithm SM2 based on elliptic curves
Part 2: Digital signature algorithm

Contents

Public key cryptographic algorithm SM2 based on elliptic curves
Part 2: Digital signature algorithm

1 Scope

This part of GM/T 0003 specifies the digital signature algorithm of the public key cryptographic algorithm SM2 based on elliptic curves, including the digital signature generation and verification algorithms, and demonstrates examples of digital signature and verification and the corresponding processes.

This part is applicable to digital signature generation and verification in commercial cryptographic applications. It meets the security requirements of identity authentication and data integrity and authenticity in many kinds of cryptographic applications. Besides, this part also provides standardization positioning and reference to product and technology for manufacturers of security products, improves the creditability and maneuverability of security products.

2 Normative references

The following parts are necessary for the application of this document. For all the references with noted dates, only the version with noted date is suitable for this document. For the references without dates, the newest version (including all the modified lists) are suitable for this part.

GM/T 0003.1–2012, Public key cryptographic algorithm SM2 based on elliptic curves — Part 1: General

3 Terms and definitions

The following terms and definitions are applicable to this document.

3.1 message

any bit string of finite length

3.2 signed message

a list of data entry comprised of a message and a digital signature of this message

3.3 signature key

the secret data reserved by the signer in the digital signature process, i.e. the secret key of the signer

3.4 signature process

the process of inputting a message, a signature secret key and system parameters of an elliptic curve and outputting a digital signature

3.5 distinguishing identifier

the information that can identify the identity of an entity without ambiguity

4 Symbols

The following symbols are applicable to this part.

A,B: two users who use the public key cryptography system

a, b: elements in F_q which define an elliptic curve E over F_q

d_A: the private key of user A

$E(F_q)$: the set of rational points on elliptic curves E over F_q (including the infinity point O)

e: the output of a cryptographic hash function on input M

e': the output of a cryptographic hash function on input M′

F_q: the finite field with q elements

G: a base point of an elliptic curve with prime order

$H_v()$: a cryptographic hash function with v bits message digest

ID_A: the distinguishing identifier of user A

M: the message to be signed

M': the message to be verified

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

n: the order of a base point G where n is a prime factor of $\#E(F_q)$

O: a special point on elliptic curve called the infinity point or zero point. it is the identity of the additive group of elliptic curve

P_A: the public key of user A

q: the number of elements of finite field F_q

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

Z_A: the hash value of the distinguishing identifier of user A, part of the system parameters of elliptic curves and the public key of user A

(r, s): the sent signature

(r', s'): the received signature

$[k]P$: a point which is k times of point P on elliptic curves, i.e. $[k]P = \underbrace{P + P + \cdots + P}_{k}$, where k is a positive integer

$[x, y]$: the 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, $\lceil 7 \rceil = 7$, $\lceil 8.3 \rceil = 9$.

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

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

5 Digital signature algorithm

5.1 General

The digital signature algorithm generates a digital signature of data by a signer and verifies the validity of the signature by a verifier. Every signer has a private key and a corresponding public key, where the private key is used to generate a signature and the public key is used by the verifier to verify the signature. Before the signature process, message $\overline{M}$ (including Z_A and M) should be compressed using a cryptographic hash function. Before the verification process, $\overline{M}'$ (including Z_A and M') should be compressed using a cryptographic hash function.

5.2 System parameters of elliptic curves

The system parameters of elliptic curves includes the size q of finite field F_q (when $q = 2^m$, also identifiers of representation of elements and reduced polynomial are involed), two element $a, b \in F_q$ which define a equation of the elliptic curves $E(F_q)$, a base point $G = (x_G, y_G)(G \neq O)$ over $E(F_q)$ where x_G and y_G are two elements of F_q, the order n of G and other optional parameters (e.g. the cofactor h of n).

The system parameters of elliptic curves and their validation shall be in line with the regulations in Clause 5 of GM/T 0003.1–2012.

5.3 User's key pair

The key pair of the user A includes the private key d_A and the public key $P_A = [d_A]G = (x_A, y_A)$.

The generation algorithm of user's key pair and the verification of public key should be in line with the regulations in Clause 6 of GM/T 0003.1–2012.

5.4 Auxiliary functions

5.4.1 Overview

Two kinds of auxiliary functions are involved in the digital signature algorithm based on elliptic curves specified in this part: cryptographic hash functions and random number generators.

5.4.2 Cryptographic hash functions

This part adopts the cryptographic hash functions approved by the State Cryptography Administration such as the SM3 cryptographic hash algorithm.

5.4.3 Random number generators

This part adopts random number generators approved by the State Cryptography Administration.

5.5 Other information of user

User A, as the signer, has a distinguishing identifier ID_A of length $entlen_A$ bits. Notation $ENTL_A$ are the two bytes converted from integer $entlen_A$. In the digital signature algorithm based on elliptic curves specified in this part, both the signer and the verifier shall use cryptographic hash functions to obtain user A's hash value Z_A. Convert the data types of the elliptic curve equation parameters a, b, the coordinates (x_G, y_G) of G and P_A's coordinates (x_A, y_A) to bit string as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012. Then $Z_A = H_{256}(ENTL_A||ID_A||a||b||x_G||y_G||x_A||y_A)$.

6 Digital signature generation algorithm and its process

6.1 Digital signature generation algorithm

Let M be the message to be signed. In order to obtain a signature (r, s) of the message M, user A as a signer should do the following:

A1: Set $\bar{M} = Z_A \parallel M$;

A2: Compute $e = H_v(\bar{M})$, and convert the type of data e to be integer as specified in Clauses 4.2.4 and 4.2.3 of GM/T 0003.1–2012;

A3: Generate a random number $k \in [1, n-1]$ using random number generators;

A4: Compute $(x_1, y_1) = [k]G$, and convert the type of data x_1 to be integer as specified in Clause 4.2.8 of GM/T 0003.1–2012;

A5: Compute $r = (e + x_1) \mod n$. If $r = 0$ or $r + k = n$, then go to A3;

A6: Compute $s = \left((1 + d_A)^{-1} \cdot (k - r \cdot d_A)\right) \mod n$. If $s = 0$, then go to A3;

A7: Convert the type of data r, s to be bit strings according to the details in Clause 4.2.2 of GM/T 0003.1–2012. Then the signature of message M is (r, s).

Note: examples of digital signature generation process, see Annex A.

6.2 Process of digital signature generation algorithm

The process of digital signature generation algorithm is depicted in Figure 1.

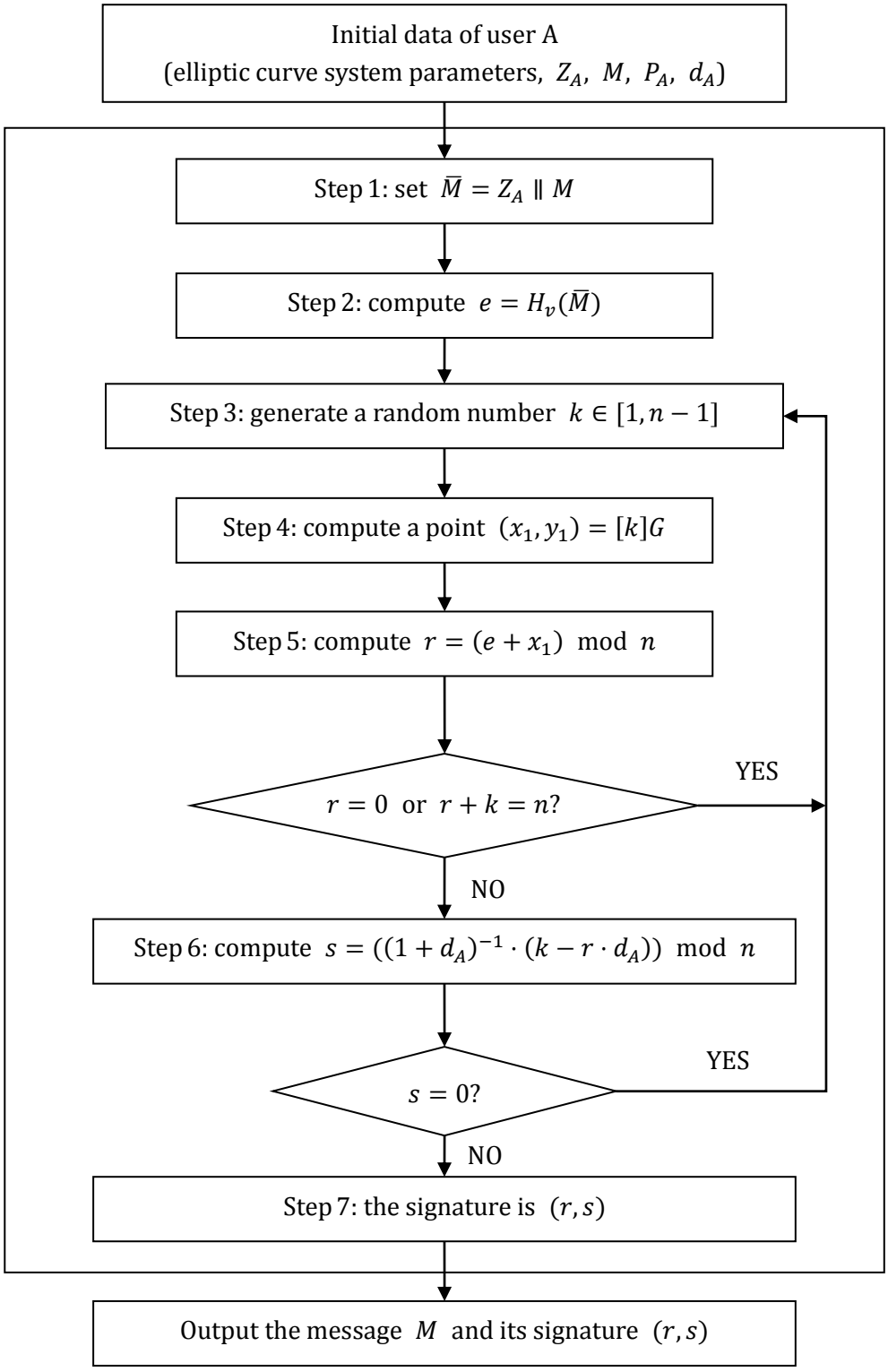


Figure 1: Digital signature generation algorithm process

7 Digital signature verification algorithm and its process

7.1 Digital signature verification algorithm

To verify the received message M' and its digital signature (r', s'), as a verifier, user B should implement the following operations:

B1: Verify whether $r' \in [1, n-1]$ holds; If not, the verifier outputs reject.

B2: Verify whether $s' \in [1, n-1]$ holds; If not, the verifier outputs reject.

B3: Set $\bar{M}' = Z_A \| M'$.

B4: Compute $e' = H_v(\bar{M}')$ and convert the type of data e' to be integer as specified in Clauses 4.2.4 and 4.2.3 of GM/T 0003.1–2012.

B5: Convert the type of data r', s' to be integers as specified in Clause 4.2.3 of GM/T 0003.1–2012 and compute $t = (r' + s') \mod n$. If $t = 0$, then the verifier output reject.

B6: Compute the point $(x_1', y_1') = [s']G + [t]P_A$.

B7: Convert the type of data x_1' to be integer with the way in Clause 4.2.8 of GM/T 0003.1–2012. Compute $R = (e' + x_1') \mod n$ and verify whether $R = r'$ holds. If so, the verifier outputs accept, otherwise outputs reject.

Note: If Z_A is not the hash function value of user A, the verifier obviously outputs reject.

For examples of digital signature verification process illustration, see Annex A.

7.2 Process of digital signature verification algorithm

The process of digital signature verification algorithm is shown in Figure 2.

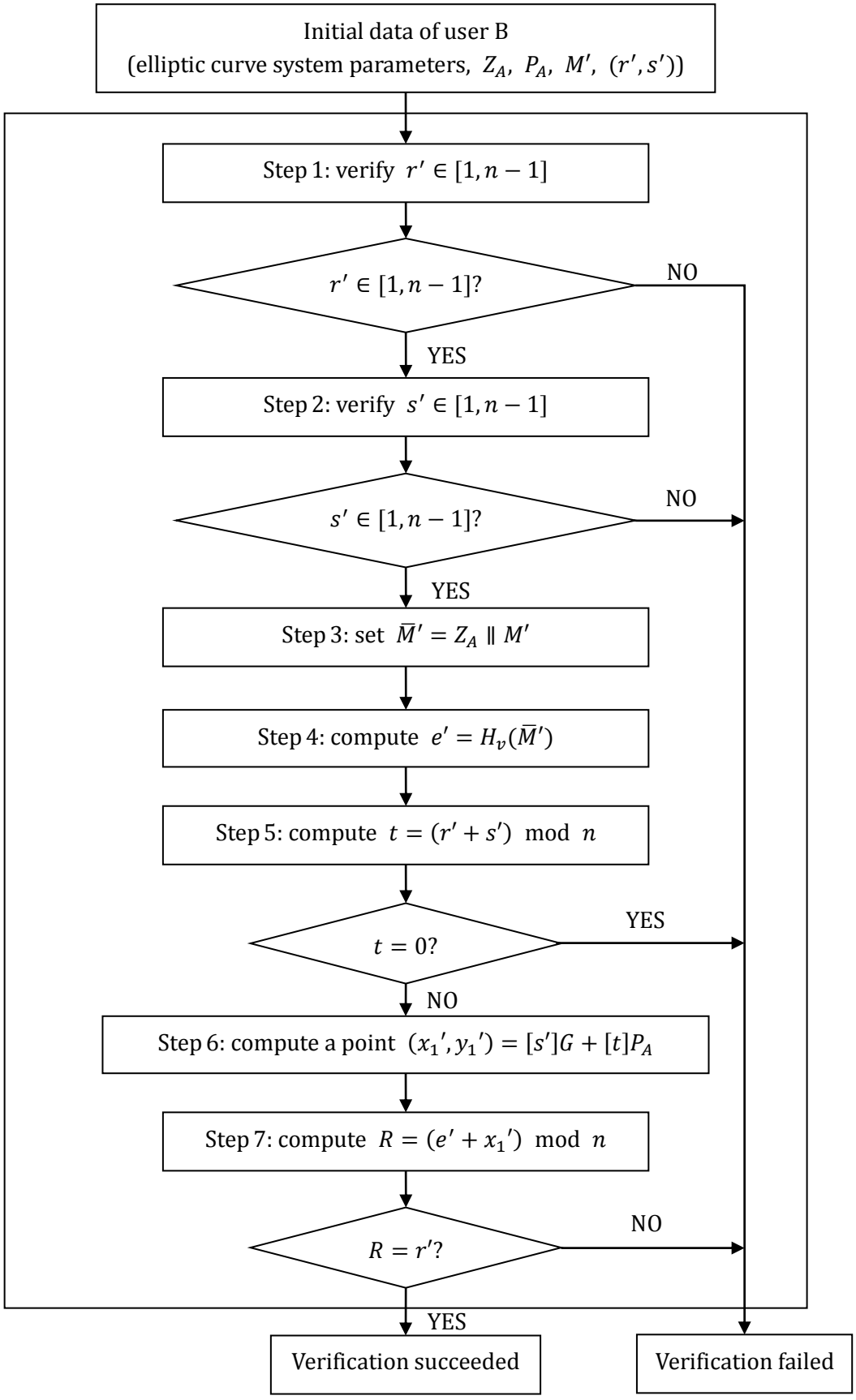


Figure 2: Digital signature verification algorithm process

Annex A
(informative)
Examples of digital signature and verification

A.1 General requirements

This annex adopts the cryptographic hash function which is specified in the GM/T 0004–2012 Cryptographic Hash Algorithm SM3, whose input is bit strings of length less than 2^{64}, and output is a hash values of length 256 bits, denoted by $H_{256}(\cdot)$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

In this annex, all messages are denoted as ASCII encoding.

Suppose user A's identity is ALICE123@YAHOO.COM. Its ASCII encoding ID_A is 414C 49434531 32334059 41484F4F 2E434F4D. $ENTL_A = 0090$.

A.2 Digital signature of elliptic curves over F_p

The elliptic curve equation is: $y^2 = x^3 + ax + b$

Example 1: $F_p - 256$

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

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

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

base point $G = (x_G, y_G)$ whose order is n

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

coordinate y_G: 0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2

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

message to be signed M: message digest

private key d_A :128B2FA8 BD433C6C 068C8D80 3DFF7979 2A519A55 171B1B65 0C23661D 15897263

public key $P_A = (x_A, y_A)$:

coordinate x_A: 0AE4C779 8AA0F119 471BEE11 825BE462 02BB79E2 A5844495

E97C04FF 4DF2548A

coordinate y_A: 7C0240F8 8F1CD4E1 6352A73C 17B7F16F 07353E53 A176D684 A9FE0C6B B798E857

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: F4A38489 E32B45B6 F876E3AC 2168CA39 2362DC8F 23459C1D 1146FC3D BFB7BC9A

Intermediate values in the steps of generating signature:

$\overline{M} = Z_A \parallel M$:

F4A38489 E32B45B6 F876E3AC 2168CA39 2362DC8F 23459C1D 1146FC3D BFB7BC9A 6D657373 61676520 64696765 7374

cryptographic hash value $e = H_{256}(\overline{M})$: B524F552 CD82B8B0 28476E00 5C377FB1 9A87E6FC 682D48BB 5D42E3D9 B9EFFE76

generate random number k: 6CB28D99 385C175C 94F94E93 4817663F C176D925 DD72B727 260DBAAE 1FB2F96F

compute point: $(x_1, y_1) = [k]G$ of the elliptic curve:

coordinate x_1: 110FCDA5 7615705D 5E7B9324 AC4B856D 23E6D918 8B2AE477 59514657 CE25D112

coordinate y_1: 1C65D68A 4A08601D F24B431E 0CAB4EBE 084772B3 817E8581 1A8510B2 DF7ECA1A

compute $r = (e + x_1) \mod n$: 40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1

$(1 + d_A)^{-1}$: 79BFCF30 52C80DA7 B939E0C6 914A18CB B2D96D85 55256E83 122743A7 D4F5F956

compute $s = (1 + d_A)^{-1} \cdot (k - r \cdot d_A) \ mod \ n$: 6FC6DAC3 2C5D5CF1 0C77DFB2 0F7C2EB6 67A45787 2FB09EC5 6327A67E C7DEEBE7

The signature of message m is (r, s):

Value r: 40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1

Value s: 6FC6DAC3 2C5D5CF1 0C77DFB2 0F7C2EB6 67A45787 2FB09EC5 6327A67E C7DEEBE7

Verify the related values:

cryptographic hash value $e' = H_{256}(\overline{M}')$: B524F552 CD82B8B0 28476E00 5C377FB1 9A87E6FC 682D48BB 5D42E3D9 B9EFFE76

compute $t = (r' + s')$ mod n: 2B75F07E D7ECE7CC C1C8986B 991F441A D324D6D6 19FE06DD 63ED32E0 C997C801

compute point $(x_0^{'}, y_0^{'}) = [s^{'}]G$ of the elliptic curve:

coordinate $x_0^{'}$: 7DEACE5F D121BC38 5A3C6317 249F413D 28C17291 A60DFD83 B835A453 92D22B0A

coordinate $y_0^{'}$: 2E49D5E5 279E5FA9 1E71FD8F 693A64A3 C4A94611 15A4FC9D 79F34EDC 8BDDEBD0

compute point $(x_{00}^{'}, y_{00}^{'}) = [t]P_A$ of the elliptic curve:

coordinate $x_{00}^{'}$: 1657FA75 BF2ADCDC 3C1F6CF0 5AB7B45E 04D3ACBE 8E4085CF A669CB25 64F17A9F

coordinate $y_{00}^{'}$: 19F0115F 21E16D2F 5C3A485F 8575A128 BBCDDF80 296A62F6 AC2EB842 DD058E50

compute point $(x_1^{'}, y_1^{'}) = [s^{'}]G + [t]P_A$ of the elliptic curve

coordinate $x_1^{'}$: 110FCDA5 7615705D 5E7B9324 AC4B856D 23E6D918 8B2AE477 59514657 CE25D112

coordinate $y_1^{'}$: 1C65D68A 4A08601D F24B431E 0CAB4EBE 084772B3 817E8581 1A8510B2 DF7ECA1A

compute $R = (e^{'} + x_1^{'})\ mod\ n$: 40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1

A.3 Digital signature of elliptic curves over F_{2^m}

The elliptic curve equation is: $y^2 + xy = x^3 + ax^2 + b$

Example 2: $F_{2^m} - 257$

generator polynomial of the base field: $x^{257} + x^{12} + 1$

coefficient a: 0

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

base point $G = (x_G, y_G)$ with order n

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

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

order n: 7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

message to be signed M: message digest

private key d_A: 771EF3DB FF5F1CDC 32B9C572 93047619 1998B2BF 7CB981D7 F5B39202 645F0931

public key $P_A = (x_A, y_A)$:

coordinate x_A: 01 65961645 281A8626 607B917F 657D7E93 82F1EA5C D931F40F 6627F357 542653B2

coordinate y_A: 01 68652213 0D590FB8 DE635D8F CA715CC6 BF3D05BE F3F75DA5 D5434544 48166612

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: 26352AF8 2EC19F20 7BBC6F94 74E11E90 CE0F7DDA CE03B27F 801817E8 97A81FD5

Intermediate values in the steps of signature generation:

$\bar{M} = Z_A \parallel M$:

26352AF8 2EC19F20 7BBC6F94 74E11E90 CE0F7DDA CE03B27F 801817E8 97A81FD5 6D657373 61676520 64696765 7374

cryptographic hash value $e = H_{256}(\bar{M})$: AD673CBD A3114171 29A9EAA5 F9AB1AA1 633AD477 18A84DFD 46C17C6F A0AA3B12

generate random number k: 36CD79FC 8E24B735 7A8A7B4A 46D454C3 97703D64 98158C60 5399B341 ADA186D6

compute point: $(x_1, y_1) = [k]G$ of the elliptic curve:

coordinate x_1: 00 3FD87D69 47A15F94 25B32EDD 39381ADF D5E71CD4 BB357E3C 6A6E0397 EEA7CD66

coordinate y_1: 00 80771114 6D73951E 9EB373A6 58214054 B7B56D1D 50B4CD6E B32ED387 A65AA6A2

compute $r = (e + x_1) \bmod n$: 6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B

$(1 + d_A)^{-1}$: 73AF2954 F951A9DF F5B4C8F7 119DAA1C 230C9BAD E60568D0 5BC3F432 1E1F4260

Compute $s = (1 + d_A)^{-1} \cdot (k - r \cdot d_A) \bmod n$: 3124C568 8D95F0A1 0252A9BE D033BEC8 4439DA38 4621B6D6 FAD77F94 B74A9556

Signature (r, s) of message M:

Value r: 6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B

Value s: 3124C568 8D95F0A1 0252A9BE D033BEC8 4439DA38 4621B6D6 FAD77F94 B74A9556

Verify the related values:

cryptographic hash value $e' = H_{256}(\bar{M}')$: AD673CBD A3114171 29A9EAA5 F9AB1AA1 633AD477 18A84DFD 46C17C6F A0AA3B12

compute $t = (r' + s') \ mod \ n$: 1E647F8F 784891A6 51AFC342 0316F44A 042D7194 4C91910F 835086C8 2CB07194

compute the point on elliptic curve $(x_0', y_0') = [s']G$:

coordinate x_0': 00 252CF6B6 3A044FCE 553EAA77 3E1E9264 44E0DAA1 0E4B8873 89D11552 EA6418F7

coordinate y_0': 00 776F3C5D B3A0D312 9EAE44E0 21C28667 92E4264B E1BEEBCA 3B8159DC A382653A

compute point $(x_{00}', y_{00}') = [t]P_A$ of the elliptic curve

coordinate x_{00}': 00 07DA3F04 0EFB9C28 1BE107EC C389F56F E76A680B B5FDEE1D D554DC11 EB477C88

coordinate y_{00}': 01 7BA2845D C65945C3 D48926C7 0C953A1A F29CE2E1 9A7EEE6B E0269FB4 803CA68B

compute point $(x_1', y_1') = [s']G + [t]P_A$ of the elliptic curve

coordinate x_1': 00 3FD87D69 47A15F94 25B32EDD 39381ADF D5E71CD4 BB357E3C 6A6E0397 EEA7CD66

coordinate y_1': 00 80771114 6D73951E 9EB373A6 58214054 B7B56D1D 50B4CD6E B32ED387 A65AA6A2

compute $R = (e' + x_1') \ mod \ n$: 6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B

Public Key cryptographic algorithm SM2 based on elliptic curves

Part 3: Key exchange protocol

Contents

Public key cryptographic algorithm SM2 based on elliptic curves

Part 3: Key exchange protocol

1 Scope

This part of GM/T 0003 specifies the key exchange protocol of public key cryptographic algorithm SM2 based on elliptic curves, and gives examples of key exchange and verification and the corresponding processes.

This part is applicable to key exchange in commercial cryptographic applications. It meets the requirements that two parties establish their shared private key (session key) by computation via two or three optional exchanges of information. Besides, this part also provides standardization location and standardization references to product and technology for manufacturers of security product, improve the creditability and maneuverability of security products.

2 Normative references

The following documents are necessary for the application of this document. For the references with noted dates, only the version on the specific date applies to this part. For the references without dates, the newest version (including all the modified lists) applies to this part.

GM/T 0003.1–2012, Public key cryptographic algorithm SM2 based on elliptic curves — Part 1: General

3 Terms and definitions

The following terms and definitions apply to this part.

3.1 key confirmation from A to B

confirmation from A to B that A holds a specific private key

3.2 key derivation function

the function that generates one or more shared secret keys on input shared secrets and other parameters known to the two parties

3.3 initiator

the user who send the first round exchanging message in the execution of a protocol

3.4 responder

the user who does not initiate the first round exchanging message in the execution of a protocol

3.5 distinguishing identifier

the information that can identify the identity of an entity without ambiguity

4 Symbols

The following symbols are applicable to this part.

A,B: Two users who use the public key cryptography system

a, b: elements in F_q which define an elliptic curve E over F_q

d_A: the private key of user A

d_B: the private key of user B

$E(F_q)$: the set of rational points on elliptic curves E over F_q (including the infinity point O)

F_q: the finite field with q elements

G: a base point of an elliptic curve with prime order

$Hash(\ )$: a cryptographic hash function

$H_v(\ )$: a cryptographic hash function with v bits message digest

h: the cofactor defined as $h = \#E(F_q)/n$, where n is the order of the base point G.

ID_A, ID_B: the distinguishing identifiers of user A and B respectively

K, K_A, K_B: the shared private keys generated in key exchange protocol

KDF(): key derivation function

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

n: the order of a base point G where n is a prime factor of $\#E(F_q)$

O: a special point on elliptic curve called the infinity point or zero point, which is the identity of the additive group of elliptic curve

P_A: the public key of user A

P_B: the public key of user B

q: the number of elements of finite field F_q

r_A: ephemeral key generated by user A in key exchange

r_B: ephemeral key generated by user B in key exchange

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

Z_A: the hash value of the distinguishing identifier of user A, part of the system parameters of elliptic curves and the public key of user A

Z_B: the hash value of the distinguishing identifier of user B, part of the system parameters of elliptic curves and the public key of user B

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

$[k]P$: a point which is k times of point P on elliptic curves, i.e. $[k]P = \underbrace{P + P + \cdots + P}_{k}$, where k is an positive integer

$[x, y]$: the 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, $\lceil 7 \rceil = 7$, $\lceil 8.3 \rceil = 9$.

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

&: the bit-wise OR operation of two integers.

5 Algorithm parameters and auxiliary functions

5.1 General

Key exchange protocol is the process of two users A and B use their respective private key and the other's public key to share a secret key, by the interactive information exchanges. The shared secret key is generally used in some symmetric cryptographic algorithm. The key exchange protocol can be used in key management and key agreement.

5.2 System parameters of elliptic curves

The system parameters of elliptic curves includes the size q of finite field F_q (when $q = 2^m$, also identifiers of representation of elements and reduced polynomial are involed), two element $a, b \in F_q$ which define the equation of the elliptic curve $E(F_q)$, a base point $G = (x_G, y_G)$ $(G \neq O)$ over $E(F_q)$ where x_G and y_G are two elements of F_q, the order n of G and other optional parameters (e.g. the cofactor h of n etc.).

The system parameters of elliptic curves and their validation shall be in line with the regulations in Clause 5 of GM/T 0003.1–2012.

5.3 User's key pair

The key pair of the user A includes the private key d_A and the public key $P_A = [d_A]G = (x_A, y_A)$. The key pair of the user B includes the private key d_B and the public key $P_B = [d_B]G = (x_B, y_B)$.

The generation algorithm of user's key pair and the verification of public key should be in line with the regulations in Clause 6 of GM/T 0003.1–2012.

5.4 Auxiliary functions

5.4.1 Overview

Three kinds of auxiliary functions are involved in the key exchange protocol based on elliptic curves specified in this part: cryptographic hash functions, key derivation functions and random number generators. These three types of auxiliary functions have directly impact on security of the key exchange protocol.

5.4.2 Cryptographic hash functions

This part adopts the cryptographic hash functions approved by the State Cryptography Administration such as the SM3 cryptographic hash algorithm.

5.4.3 Key derivation function

The functionality of key derivation functions is to derive key data from a shared secret bit string. In the process of key agreement, on input the shared secret bit string obtained by key exchange protocol, the key derivation function outputs a required session key or a key data required by further encryption.

Key derivation function needs to invoke the cryptographic hash function.

Let $H_v()$ be a cryptographic hash function which outputs a hash value of length v bits.

Key derivation function $KDF(Z, klen)$ is defined as follows:

Input: a bit string Z, an integer $klen$ which represents the bit length of the resulting secret key data and is required to be smaller than $(2^{32} - 1)v$.

Output: the secret key bit string K of length $klen$ bits.

a) Initialize a 32-bit counter ct = 0x00000001;

b) For i from 1 to $\lceil \frac{klen}{v} \rceil$ do:

b.1) compute $Ha_i = H_v(Z \| ct)$;

b.2) $ct++$;

c) If $\frac{klen}{v}$ is an integer, let $Ha!_{\lceil \frac{klen}{v} \rceil} = Ha_{\lceil \frac{klen}{v} \rceil}$; Otherwise let $Ha!_{\lceil \frac{klen}{v} \rceil}$ be the leftmost $(klen - (v \times \lfloor \frac{klen}{v} \rfloor))$ bits of $Ha_{\lceil \frac{klen}{v} \rceil}$.

d) Let $K = Ha_1 \| \cdots \| Ha_{\lceil \frac{klen}{v} \rceil - 1} \| Ha!_{\lceil \frac{klen}{v} \rceil}$.

5.4.4 Random number generators

This part adopts random number generators approved by the State Cryptography Administration.

5.5 Other information of user

User A has a distinguishing identifier ID_A of length $entlen_A$ bits. Notation $ENTL_A$ are the two bytes converted from integer $entlen_A$. User B has a distinguishing identifier ID_B of length $entlen_B$ bits. Notation $ENTL_B$ are the two bytes converted from integer $entlen_B$. In the key exchange protocol based on elliptic curves specified in this part, both parties A and B participated in the key agreement use the cryptographic hash function to get their respective hash values Z_A and Z_B. Convert the data types of the elliptic curve equation parameters a, b, coordinates (x_G, y_G) of G, coordinates (x_A, y_A) of P_A and coordinates (x_B, y_B) of P_B to bit strings as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012, then $Z_A = H_{256}(ENTL_A||ID_A||a||b||x_G||y_G||x_A||y_A)$ and $Z_B = H_{256}(ENTL_B||ID_B||a||b||x_G||y_G||x_B||y_B)$.

6 Key exchange protocol and the process

6.1 Key exchange protocol

Suppose the length of the secret key established by A and B is $klen$. A is the initiator and B is the responder. In order to get the same secret key, user A and B should perform the following operations:

Let $w = \lceil(\lceil\log_2(n)\rceil/2)\rceil - 1$.

User A:

A1: Generate a random number $r_A \in [1, n-1]$ with random number generators;

A2: Compute the point $R_A = [r_A]G = (x_1, y_1)$ on elliptic curve;

A3: Sent R_A to user B;

User B:

B1: Generate a random number $r_B \in [1, n-1]$ with random number generators;

B2: Compute the point $R_B = [r_B]G = (x_2, y_2)$ of the elliptic curve;

B3: Choose an element x_2 from R_B, convert its data type to an integer as specified in Clause 4.2.8 of GM/T0003.1–2012, and compute $\bar{x}_2 = 2^w + (x_2 \& (2^w - 1))$;

B4: Compute $t_B = (d_B + \bar{x}_2 \cdot r_B) \bmod n$;

B5: Verify whether R_A satisfies the elliptic curve equation. If not, the agreement is failed. Otherwise, choose an element x_1 from R_A, compute $\bar{x}_1 = 2^w + (x_1 \& (2^w - 1))$ and then convert its data type to an integer as specified in Clause 4.2.8 of GM/T0003.1–2012;

B6: Compute the point $V = [h \cdot t_B](P_A + [\bar{x}_1]R_A) = (x_V, y_V)$ of the elliptic curve. If V is the infinity point, the agreement is failed; Otherwise, convert the types of data x_V, y_V to bit strings as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012;

B7: Compute $K_B = KDF(x_V \parallel y_V \parallel Z_A \parallel Z_B, klen)$;

B8: (Optional) Convert the data type of R_A's coordinates x_1, y_1 and R_B's coordinates x_2, y_2 to bit strings as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012, and compute $S_B = Hash(0x02 \parallel y_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$;

B9: Send R_B and optionally S_B to user A;

User A:

A4: choose an element x_1 from R_A, and compute $\bar{x}_1 = 2^w + (x_1 \& (2^w - 1))$ by converting its data type to integer as specified in Clause 4.2.8 of GM/T0003.1–2012;

A5: Compute $t_A = (d_A + \bar{x}_1 \cdot r_A) \bmod n$;

A6: Verify whether R_B satisfies the elliptic curve equation. If not, the agreement is failed. Otherwise, Choose an element x_2 from R_B, and compute $\bar{x}_2 = 2^w + (x_2 \& (2^w - 1))$ by converting its data type to integer as specified in Clause 4.2.8 of GM/T0003.1–2012;

A7: Compute the point $U = [h \cdot t_A](P_B + [\bar{x}_2]R_B) = (x_U, y_U)$ of the elliptic curve. If U is the infinity point, the agreement is failed; Otherwise convert the data type of x_U, y_U to integers as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012;

A8: Compute $K_A = KDF(x_U \parallel y_U \parallel Z_A \parallel Z_B, klen)$;

A9: (Optional) Convert the data type of R_A's coordinates x_1, y_1 and R_B's coordinates x_2, y_2 to bit strings as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012, compute $S_1 = Hash(0x02 \parallel y_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$ and verify whether $S_1 = S_B$ holds; If not, key confirmation from B to A is failed;

A10: (Optional) Compute $S_A = Hash(0x03 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$ and send S_A to user B;

User B:

B10: (Optional) Compute $S_2 = Hash(0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$ and verify whether $S_2 = S_A$ holds. If not, key confirmation from A to B is failed;

Note: If Z_A, Z_B are not the corresponding hash values of user A and B respectively, they obviously cannot achieve a consistent shared secret key. The examples of the process of key exchange protocol are described in Annex A.

6.2 Process of key exchange protocol

The process of the key exchange protocol is depicted in Figure 1.

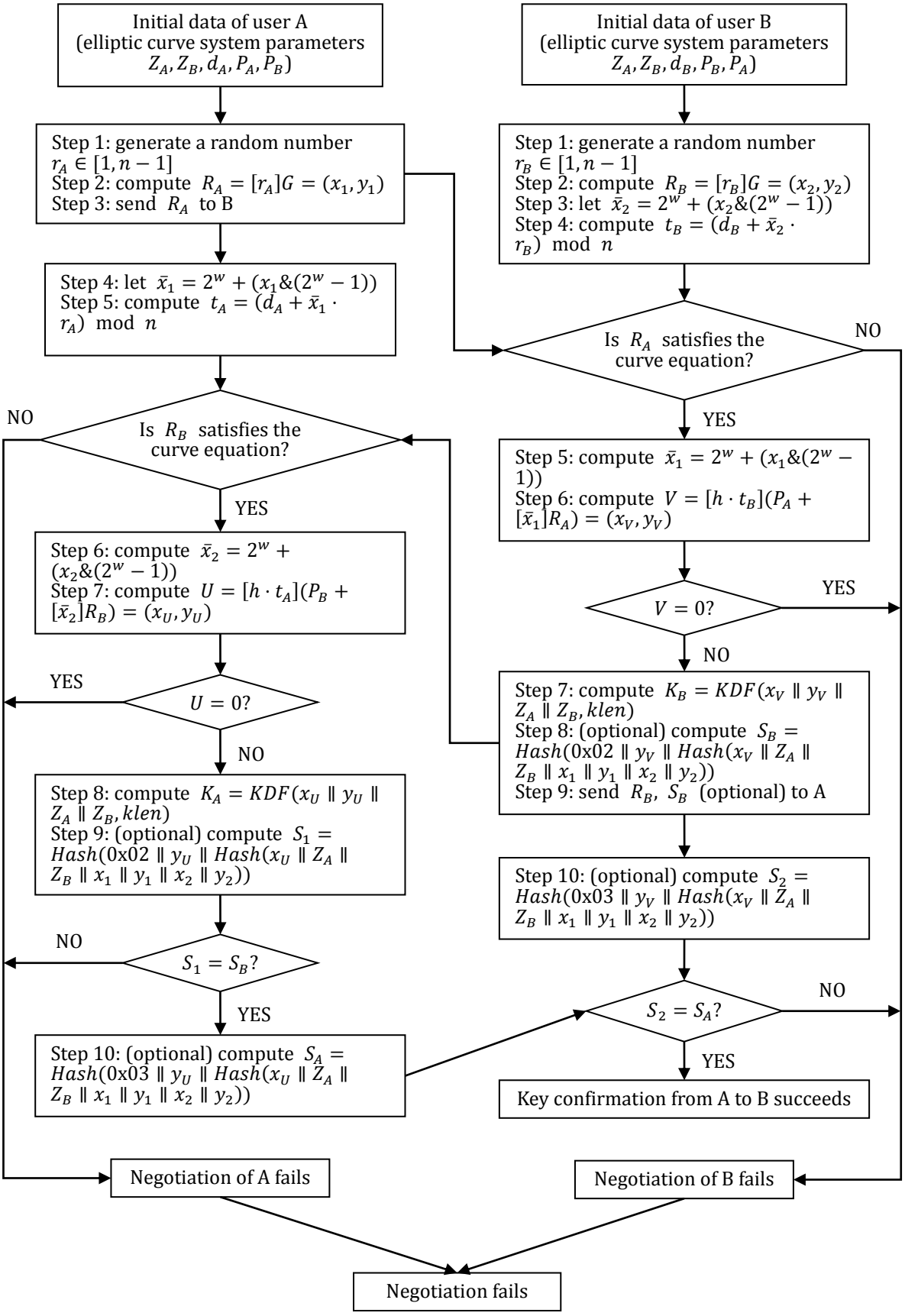


Figure 1: Key exchange protocol process

Annex A
(informative)
Examples of key exchange and verification

A.1 General requirements

This annex adopts the cryptographic hash function specified in GM/T 0004−2012 SM3 Cryptographic Hash Algorithm, whose input is a bit string of length less than 2^{64}, and output is a hash value of length 256 bits, denoted $H_{256}()$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

Suppose user A's identity is ALICE123@YAHOO.COM. Its ASCII encoding $\mathrm{ID_A}$ is 414C 49434531 32334059 41484F4F 2E434F4D. $\mathrm{ENTL_A} = 0090$.

Suppose user B's identity is BILL456@YAHOO.COM. Its ASCII encoding $\mathrm{ID_B}$ is: 42 494C4C34 35364059 41484F4F 2E434F4D. $\mathrm{ENTL_B} = 0088$.

A.2 Key exchange protocol on elliptic curves over F_p

The equation of elliptic curve is: $y^2 = x^3 + ax + b$

Example 1: $F_p - 256$

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

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

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

cofactor h: 1

Base point $G = (x_G, y_G)$ whose order is n

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

coordinate y_G: 0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2

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

user A's private key d_A: 6FCBA2EF 9AE0AB90 2BC3BDE3 FF915D44 BA4CC78F 88E2F8E7 F8996D3B 8CCEEDEE

user A's public key $P_A = (x_A, y_A)$:

coordinate x_A: 3099093B F3C137D8 FCBBCDF4 A2AE50F3 B0F216C3 122D7942 5FE03A45 DBFE1655

coordinate y_A: 3DF79E8D AC1CF0EC BAA2F2B4 9D51A4B3 87F2EFAF 48233908 6A27A8E0 5BAED98B

user B's private key d_B: 5E35D7D3 F3C54DBA C72E6181 9E730B01 9A84208C A3A35E4C 2E353DFC CB2A3B53

user B's public key $P_B = (x_B, y_B)$:

coordinate x_B: 245493D4 46C38D8C C0F11837 4690E7DF 633A8A4B FB3329B5 ECE604B2 B4F37F43

coordinate y_B: 53C0869F 4B9E1777 3DE68FEC 45E14904 E0DEA45B F6CECF99 18C85EA0 47C60A4C

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31

hash value $Z_B = H_{256}(ENTL_B \parallel ID_B \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_B \parallel y_B)$

Z_B: 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67

Related values in step A1-A3 in the key exchange protocol:

generate random number r_A: 83A2C9C8 B96E5AF7 0BD480B4 72409A9A 327257F1 EBB73F5B 073354B2 48668563

compute point $R_A = [r_A]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0

coordinate y_1: 0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A

Related values in step B1-B9 in the key exchange protocol:

generate random number r_B: 33FE2194 0342161C 55619C4A 0C060293 D543C80A F19748CE 176D8347 7DE71C80

compute point $R_B = [r_B]G = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: 1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5

coordinate y_2: 54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4

take $\bar{x}_2 = 2^{127} + (x_2 \& (2^{127} - 1))$: B8F2B533 7B3DCF45 14E8BBC1 9D900EE5

compute $t_B = (d_B + \bar{x}_2 \cdot r_B) \ mod \ n$: 2B2E11CB F03641FC 3D939262 FC0B652A 70ACAA25 B5369AD3 8B375C02 65490C9F

take $\bar{x}_1 = 2^{127} + (x_1 \& (2^{127} - 1))$: E856C095 05324A6D 23150C40 8F162BF0

Compute the point $[\bar{x}_1]R_A = (x_{A0}, y_{A0})$ of the elliptic curve:

coordinate x_{A0}: 2079015F 1A2A3C13 2B67CA90 75BB2803 1D6F2239 8DD8331E 72529555 204B495B

coordinate y_{A0}: 6B3FE6FB 0F5D5664 DCA16128 B5E7FCFD AFA5456C 1E5A914D 1300DB61 F37888ED

compute point $P_A + [\bar{x}_1]R_A = (x_{A1}, y_{A1})$ of the elliptic curve:

coordinate x_{A1}: 1C006A3B FF97C651 B7F70D0D E0FC09D2 3AA2BE7A 8E9FF7DA F32673B4 16349B92

coordinate y_{A1}: 5DC74F8A CC114FC6 F1A75CB2 86864F34 7F9B2CF2 9326A270 79B7D37A FC1C145B

compute $V = [h \cdot t_B](P_A + [\bar{x}_1]R_A) = (x_V, y_V)$:

coordinate x_V: 47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905

coordinate y_V: 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295

compute $K_B = KDF(x_V \ \| \ y_V \ \| \ Z_A \ \| \ Z_B, klen)$:

$x_V \ \| \ y_V \ \| \ Z_A \ \| \ Z_B$:

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67

$klen = 128$

shared secret key K_B: 55B0AC62 A6B927BA 23703832 C853DED4

compute optional term $S_B = Hash(0x02 \ \| \ y_V \ \| \ Hash(x_V \ \| \ Z_A \ \| \ Z_B \ \| \ x_1 \ \| \ y_1 \ \| \ x_2 \ \| \ y_2))$:

$x_V \ \| \ Z_A \ \| \ Z_B \ \| \ x_1 \ \| \ y_1 \ \| \ x_2 \ \| \ y_2$:

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31

6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67 6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0 0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A 1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5 54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

0x02 $\parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

02 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

optional term S_B: 284C8F19 8F141B50 2E81250F 1581C7E9 EEB4CA69 90F9E02D F388B454 71F5BC5C

Related values in steps A4-A10 in the key exchange protocol:

take $\bar{x}_1 = 2^{127} + (x_1 \& (2^{127} - 1))$: E856C095 05324A6D 23150C40 8F162BF0

compute $t_A = (d_A + \bar{x}_1 \cdot r_A) \ mod \ n$: 236CF0C7 A177C65C 7D55E12D 361F7A6C 174A7869 8AC099C0 874AD065 8A4743DC

take $\bar{x}_2 = 2^{127} + (x_2 \& (2^{127} - 1))$: B8F2B533 7B3DCF45 14E8BBC1 9D900EE5

compute point $[\bar{x}_2]R_B = (x_{B0}, y_{B0})$ of the elliptic curve:

coordinate x_{B0}: 66864274 6BFC066A 1E731ECF FF51131B DC81CF60 9701CB8C 657B25BF 55B7015D

coordinate y_{B0}: 1988A7C6 81CE1B50 9AC69F49 D72AE60E 8B71DB6C E087AF84 99FEEF4C CD523064

compute point $P_B + [\bar{x}_2]R_B = (x_{B1}, y_{B1})$ of the elliptic cuve:

coordinate x_{B1}: 7D2B4435 10886AD7 CA3911CF 2019EC07 078AFF11 6E0FC409 A9F75A39 01F306CD

coordinate y_{B1}: 331F0C6C 0FE08D40 5FFEDB30 7BC255D6 8198653B DCA68B9C BA100E73 197E5D24

compute $U = [h \cdot t_A](P_B + [\bar{x}_2]R_B) = (x_U, y_U)$:

coordinate x_U: 47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905

coordinate y_U: 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295

compute $K_A = KDF(x_U \parallel y_U \parallel Z_A \parallel Z_B, klen)$:

$x_U \parallel y_U \parallel Z_A \parallel Z_B$:

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67

$klen = 128$

shared secret key K_A: 55B0AC62 A6B927BA 23703832 C853DED4

compute optional term $S_1 = Hash(0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67 6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0 0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A 1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5 54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

$0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

02 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

optional term S_1: 284C8F19 8F141B50 2E81250F 1581C7E9 EEB4CA69 90F9E02D F388B454 71F5BC5C

compute optional term $S_A = Hash(0x03 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$):

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 336E6656 119ABD67 6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0 0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A 1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5 54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$: FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

0x03 $\parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

optional term S_A: 23444DAF 8ED75343 66CB901C 84B3BDBB 63504F40 65C1116C 91A4C006 97E6CF7A

Related values in step B10 in the key exchange protocol:

compute optional term $S_2 = Hash(0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$):

47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905 E4D1D0C3 CA4C7F11 BC8FF8CB 3F4C02A7 8F108FA0 98E51A66 8487240F 75E20F31 6B4B6D0E 276691BD 4A11BF72 F4FB501A E309FDAC B72FA6CC 6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0 0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A 1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5 54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

$0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295 FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647

optional term S_2: 23444DAF 8ED75343 66CB901C 84B3BDBB 63504F40 65C1116C 91A4C006 97E6CF7A

A.3 Key exchange protocol on elliptic curves over F_{2^m}

elliptic curve equation: $y^2 + xy = x^3 + ax^2 + b$

Example 2: $F_{2^m} - 257$

generator polynomial of the base field: $x^{257} + x^{12} + 1$

coefficient a: 0

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

cofactor h: 4

base point: $G = (x_G, y_G)$ with order n

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

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

order n: 7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

user A's private key d_A: 4813903D 254F2C20 A94BC570 42384969 54BB5279 F861952E F2C5298E 84D2CEAA

user A's public key $P_A = (x_A, y_A)$:

coordinate x_A: 00 8E3BDB2E 11F91933 88F1F901 CCC857BF 49CFC065 FB38B906 9CAAE6D5 AFC3592F

coordinate y_A: 00 4555122A AC0075F4 2E0A8BBD 2C0665C7 89120DF1 9D77B4E3 EE4712F5 98040415

user B's private key d_B: 08F41BAE 0922F47C 212803FE 681AD52B 9BF28A35 E1CD0EC2 73A2CF81 3E8FD1DC

user B's public key $P_B = (x_B, y_B)$:

coordinate x_B: 00 34297DD8 3AB14D5B 393B6712 F32B2F2E 938D4690 B095424B 89DA880C 52D4A7D9

coordinate y_B: 01 99BBF11A C95A0EA3 4BBD00CA 50B93EC2 4ACB6833 5D20BA5D CFE3B33B DBD2B62D

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: ECF00802 15977B2E 5D6D61B9 8A99442F 03E8803D C39E349F 8DCA5621 A9ACDF2B

hash value $Z_B = H_{256}(ENTL_B \parallel ID_B \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_B \parallel y_B)$:

Z_B: 557BAD30 E183559A EEC3B225 6E1C7C11 F870D22B 165D015A CF9465B0 9B87B527

Related values in step A1-A3 in the key exchange protocol:

generate random number r_A: 54A3D667 3FF3A6BD 6B02EBB1 64C2A3AF 6D4A4906 229D9BFC E68CC366 A2E64BA4

compute point $R_A = [r_A]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 01 81076543 ED19058C 38B313D7 39921D46 B80094D9 61A13673 D4A5CF8C 7159E304

coordinate y_1: 01 D8CFFF7C A27A01A2 E88C1867 3748FDE9 A74C1F9B 45646ECA 0997293C 15C34DD8

Related values in step B1-B9 in the key exchange protocol:

generate random number r_B: 1F219333 87BEF781 D0A8F7FD 708C5AE0 A56EE3F4 23DBC2FE 5BDF6F06 8C53F7AD

compute point $R_B = [r_B]G = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: 00 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A

coordinate y_2: 00 16599BB5 2ED9EAFA D01CFA45 3CF3052E D60184D2 EECFD42B 52DB7411 0B984C23

take $\bar{x}_2 = 2^{127} + (x_2 \& (2^{127} - 1))$: E8CC43FF A5F2623B 9BD04E46 8D322A2A

compute $t_B = (d_B + \bar{x}_2 \cdot r_B) \ mod \ n$: 3D51D331 14A453A0 5791DB63 5B45F8DB C54686D7 E2212D49 E4A717C6 B10DEDB0

compute $h \cdot t_B \ mod \ n$: 75474CC4 52914E81 5E476D8D 6D17E36F 5882EE67 A1CDBC26 FE4122B0 B741A0A3

take $\bar{x}_1 = 2^{127} + (x_1 \& (2^{127} - 1))$: B80094D9 61A13673 D4A5CF8C 7159E304

compute point $[\bar{x}_1]R_A = (x_{A0}; y_{A0})$ of the elliptic curve:

coordinate x_{A0}: 01 98AB5F14 349B6A46 F77FBFCB DDBFCD34 320DC1F4 C546D13C 3A9F0E83 0C39B579

coordinate y_{A0}: 00 BFB49224 ACCE2E51 04CD4519 C0CBE3AD 0C19BF11 805BE108 59069AA6 9317A2B7

compute point $P_A + [\bar{x}_1]R_A = (x_{A1}, y_{A1})$ of the elliptic curve:

coordinate x_{A1}: 00 24A92F64 66A37C5C 12A2C68D 58BFB0F0 32F2B976 60957CB0 5E63F961 F160FE57

coordinate y_{A1}: 00 F74A4F17 DC560A55 FDE0F1AB 168BCBF7 6502E240 BA2D6BD6 BE6E5D79 16B288FC

compute $V = [h \cdot t_B](P_A + [\bar{x}_1]R_A) = (x_V, y_V)$:

coordinate x_V : 00 DADD0874 06221D65 7BC3FA79 FF329BB0 22E9CB7D DFCFCCFE 277BE8CD 4AE9B954

coordinate y_V: 01 F0464B1E 81684E5E D6EF281B 55624EF4 6CAA3B2D 37484372 D91610B6 98252CC9

compute $K_B = KDF(x_V \parallel y_V \parallel Z_A \parallel Z_B, klen)$:

$x_V \parallel y_V \parallel Z_A \parallel Z_B$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 5401F046 4B1E8168 4E5ED6EF 281B5562 4EF46CAA 3B2D3748 4372D916 10B69825 2CC9ECF0 08021597 7B2E5D6D 61B98A99 442F03E8 803DC39E 349F8DCA 5621A9AC DF2B557B AD30E183 559AEEC3 B2256E1C 7C11F870 D22B165D 015ACF94 65B09B87 B527

$klen = 128$

shared secret key K_B: 4E587E5C 66634F22 D973A7D9 8BF8BE23

compute optional term $S_B = Hash(0x02 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 54ECF008 0215977B 2E5D6D61 B98A9944 2F03E880 3DC39E34 9F8DCA56 21A9ACDF 2B557BAD 30E18355 9AEEC3B2 256E1C7C 11F870D2 2B165D01 5ACF9465 B09B87B5 27018107 6543ED19 058C38B3 13D73992 1D46B800 94D961A1 3673D4A5 CF8C7159 E30401D8 CFFF7CA2 7A01A2E8 8C186737 48FDE9A7 4C1F9B45 646ECA09 97293C15 C34DD800 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A 0016599B B52ED9EA FAD01CFA 453CF305 2ED60184 D2EECFD4 2B52DB74 110B984C 23

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C

0x02 $\parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

02 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C

optional term S_B: 4EB47D28 AD3906D6 244D01E0 F6AEC73B 0B51DE15 74C13798 184E4833 DBAE295A

Related values in step A4-A10 in the key exchange protocol:

take $\bar{x}_1 = 2^{127} + (x_1 \& (2^{127} - 1))$: B80094D9 61A13673 D4A5CF8C 7159E304

compute $t_A = (d_A + \bar{x}_1 \cdot r_A) \ mod \ n$: 18A1C649 B94044DF 16DC8634 993F1A4A EE3F6426 DFE14AC1 3644306A A5A94187

compute $h \cdot t_A \ mod \ n$: 62871926 E501137C 5B7218D2 64FC692B B8FD909B 7F852B04 D910C1AA 96A5061C

take $\bar{x}_2 = 2^{127} + (x_2 \& (2^{127} - 1))$: E8CC43FF A5F2623B 9BD04E46 8D322A2A

compute point $[\bar{x}_2]R_B = (x_{B0}, y_{B0})$ of the elliptic curve

coordinate x_{B0}: 01 0AA3BAC9 7786B629 22F93414 57AC64F7 2552AA15 D9321677 A10C7021 33B16735

coordinate y_{B0}: 00 C10837F4 8F53C46B 714BCFBF AA1AD627 11FCB03C 0C25B366 BF176A2D C7B8E62E

compute point $P_B + [\bar{x}_2]R_B = (x_{B1}, y_{B1})$ of the elliptic curve:

coordinate x_{B1}: 00 C7A446E1 98DB4278 60C3BB50 ED2197DE B8161973 9141CA61 03745035 9FAD9A99

coordinate y_{B1}: 00 602E5A42 17427EAB C5E3917D E81BFFA1 D806591A F949DD7C 97EF90FD 4CF0A42D

compute $U = [h \cdot t_A](P_B + [\bar{x}_2]R_B) = (x_U, y_U)$:

coordinate x_U: 00 DADD0874 06221D65 7BC3FA79 FF329BB0 22E9CB7D DFCFCCFE 277BE8CD 4AE9B954

coordinate y_U: 01 F0464B1E 81684E5E D6EF281B 55624EF4 6CAA3B2D 37484372 D91610B6 98252CC9

compute $K_A = KDF(x_U \parallel y_U \parallel Z_A \parallel Z_B; klen)$:

$x_U \parallel y_U \parallel Z_A \parallel Z_B$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 5401F046 4B1E8168 4E5ED6EF 281B5562 4EF46CAA 3B2D3748 4372D916 10B69825 2CC9ECF0 08021597 7B2E5D6D 61B98A99 442F03E8 803DC39E 349F8DCA 5621A9AC DF2B557B AD30E183 559AEEC3 B2256E1C 7C11F870 D22B165D 015ACF94 65B09B87 B527

$klen = 128$

shared secret key K_A: 4E587E5C 66634F22 D973A7D9 8BF8BE23

compute optional term $S_1 = Hash(0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 54ECF008 0215977B 2E5D6D61 B98A9944 2F03E880 3DC39E34 9F8DCA56 21A9ACDF 2B557BAD 30E18355 9AEEC3B2 256E1C7C 11F870D2 2B165D01 5ACF9465 B09B87B5 27018107 6543ED19 058C38B3 13D73992 1D46B800 94D961A1 3673D4A5 CF8C7159 E30401D8 CFFF7CA2 7A01A2E8 8C186737 48FDE9A7 4C1F9B45 646ECA09 97293C15 C34DD800 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A 0016599B B52ED9EA FAD01CFA 453CF305 2ED60184 D2EECFD4 2B52DB74 110B984C 23

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C

$0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

02 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C

optional term S_1: 4EB47D28 AD3906D6 244D01E0 F6AEC73B 0B51DE15 74C13798 184E4833 DBAE295A

compute optional term $S_A = Hash(0x03 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 54ECF008 0215977B 2E5D6D61 B98A9944 2F03E880 3DC39E34 9F8DCA56 21A9ACDF 2B557BAD 30E18355 9AEEC3B2 256E1C7C 11F870D2

2B165D01 5ACF9465 B09B87B5 27018107 6543ED19 058C38B3 13D73992 1D46B800 94D961A1 3673D4A5 CF8C7159 E30401D8 CFFF7CA2 7A01A2E8 8C186737 48FDE9A7 4C1F9B45 646ECA09 97293C15 C34DD800 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A 0016599B B52ED9EA FAD01CFA 453CF305 2ED60184 D2EECFD4 2B52DB74 110B984C 23

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$: E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C

0x03 $\parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C

ptional term S_A: 588AA670 64F24DC2 7CCAA1FA B7E27DFF 811D500A D7EF2FB8 F69DDF48 CC0FECB7

Related values in step B10 in the key exchange protocol:

compute optional term $S_2 = Hash(0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

00DADD08 7406221D 657BC3FA 79FF329B B022E9CB 7DDFCFCC FE277BE8 CD4AE9B9 54ECF008 0215977B 2E5D6D61 B98A9944 2F03E880 3DC39E34 9F8DCA56 21A9ACDF 2B557BAD 30E18355 9AEEC3B2 256E1C7C 11F870D2 2B165D01 5ACF9465 B09B87B5 27018107 6543ED19 058C38B3 13D73992 1D46B800 94D961A1 3673D4A5 CF8C7159 E30401D8 CFFF7CA2 7A01A2E8 8C186737 48FDE9A7 4C1F9B45 646ECA09 97293C15 C34DD800 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A 0016599B B52ED9EA FAD01CFA 453CF305 2ED60184 D2EECFD4 2B52DB74 110B984C 23

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$: E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C

0x03 $\parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C

optional term S_2: 588AA670 64F24DC2 7CCAA1FA B7E27DFF 811D500A D7EF2FB8 F69DDF48 CC0FECB7

Public key cryptographic algorithm SM2 based on elliptic curves
Part 4: Public key encryption algorithm

Content

Public key cryptographic algorithm SM2 based on elliptic curves
Part 4: Public key encryption algorithm

1 Scope

This part of GM/T 0003 specifies the public encryption algorithm of public key cryptographic algorithm SM2 based on elliptic curves, and gives examples of encryption and decryption of messages and the corresponding processes.

This part is applicable to encryption and decryption of messages in commercial cryptographic applications. A sender of messages can encrypt the messages using the public key of a receiver, while the receiver can make decryption with his corresponding private key to get the messages. Besides, this part also provides standardization positioning and standardization references to product and technology for manufacturers of security product, improve the creditability and maneuverability of security products.

2 Normative references

The following documents are necessary for the application of this document. For the references with noted dates, only the version on the specific date applies to this part. For the references without dates, the newest version (including all the modified lists) applies to this part.

GM/T 0003.1–2012, Public key cryptographic algorithm SM2 based on elliptic curves — Part 1: General

3 Terms and definitions

The following terms and definitions apply to this part.

3.1 secret key

a kind of key, which is shared between the sender and the receiver while unknown to the third party

3.2 message

any bit string of finite length

3.3 key derivation function

the function that generates one or more shared secret keys on input shared secrets and other parameters known to the two parties

4 Symbols

The following symbols are applicable to this part.

A, B: two users who use the public key cryptography system

a, b: elements in F_q which define an elliptic curve E over F_q

d_A: the private key of user A.

d_B: the private key of user B

$E(F_q)$: the set of rational points on elliptic curves E over F_q (including the infinity point O).

F_q: the finite field with q elements.

G: a base point of an elliptic curve with prime order

$Hash()$: a cryptographic hash function.

$H_v( )$: a cryptographic hash function with v bits message digest

$KDF()$: key derivation function

M: the message to be encrypted

M': the message obtained by decryption

n: the order of a base point G where n is a prime factor of $\#E(F_q)$

O: a special point on elliptic curve called the infinity point or zero point. it is the identity of the additive group of elliptic curve.

P_B: public key of user B

q: the number of elements of finite field F_q

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

$[k]P$: a point which is k times of point P on elliptic curves, i.e. $[k]P = \underbrace{P + P + \cdots + P}_{k\ P's}$, where k is an positive integer

$[x, y]$: the 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, $\lceil 7 \rceil = 7$, $\lceil 8.3 \rceil = 9$.

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

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

5 Algorithm parameters and auxiliary functions

5.1 General

Public key encryption algorithm stipulates that a sender generates a ciphertext by encrypting a message with a receiver's public key, while the receiver can get the original message by decrypting the received ciphertext with his own private key.

5.2 System parameters of elliptic curves

The system parameters of elliptic curves includes the size q of finite field F_q (when $q = 2^m$, also identifiers of representation of elements and reduced polynomial are involved), two element $a, b \in F_q$ which define a equation of the elliptic curve $E(F_q)$, a base point $G = (x_G, y_G)$ $(G \neq O)$ over $E(F_q)$ where x_G and y_G are two elements of F_q, the order n of G and other optional parameters (e.g. the cofactor h of n etc.).

The system parameters of elliptic curves and their validation shall be in line with the regulations in Clause 5 of GM/T 0003.1–2012.

5.3 User's key pair

The key pair of B consists of the private key d_B and the public key $P_B = [d_B]G$.

The generation algorithm of user's key pairs and the validation algorithm of public keys should be consistent with the specification in Clause 6 of GM/T 003.1–2012.

5.4 Auxiliary functions

5.4.1 Overview

Three kinds of auxiliary functions are involved in the public key encryption algorithm based on elliptic curves specified in this part: cryptographic hash functions, key derivation functions and pseudo-random number generators. The strength of these three types of auxiliary functions has directly impact on security of the encryption algorithm.

5.4.2 Cryptographic hash function

This part adopts the cryptographic hash functions approved by the State Cryptography Administration such as the SM3 cryptographic hash algorithm.

5.4.3 Key derivation function

The functionality of key derivation functions is to derive key data from a shared secret bit string. In the process of key agreement, on input the shared secret bit string obtained

by key exchange protocol, the key derivation function outputs a required session key or a key data required by further encryption.

Key derivation function needs to invoke the cryptographic hash function.

Let $H_v()$ be a cryptographic hash function which outputs a hash value of length v bits.

Key derivation function $KDF(Z, klen)$ is defined as follows:

Input: a bit string Z, an integer klen which represents the bit length of the resulting secret key data and is required to be smaller than $(2^{32} - 1)v$.

Output: the secret key bit string K of length $klen$.

a) Initialize a 32-bit counter ct = 0x00000001;

b) For i from 1 to $\left\lceil \frac{klen}{v} \right\rceil$ do:

b.1) compute $Ha_i = H_v(Z \parallel ct)$;

b.2) $ct++$;

c) If $\frac{klen}{v}$ is an integer, let $Ha!_{\left\lceil \frac{klen}{v} \right\rceil} = Ha_{\left\lceil \frac{klen}{v} \right\rceil}$; Otherwise let $Ha!_{\left\lceil \frac{klen}{v} \right\rceil}$ be the leftmost $(klen - (v \times \left\lfloor \frac{klen}{v} \right\rfloor))$ bits of $Ha_{\left\lceil \frac{klen}{v} \right\rceil}$.

d) Let $K = Ha_1 \parallel \cdots \parallel Ha_{\left\lceil \frac{klen}{v} \right\rceil - 1} \parallel Ha!_{\left\lceil \frac{klen}{v} \right\rceil}$.

5.4.4 Random number generators

This part adopts random number generators approved by the State Cryptography Administration.

6 Encryption algorithm and its process

6.1 Encryption algorithm

Suppose the message to be sent is the bit string M and $klen$ represents the bit-length of M.

To encrypt the plaintext M, encryption user A should perform the following procedures:

A1: generate a random number $k \in [1, n-1]$ with the random number generator.

A2: compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve, and convert the data type of C_1 to bit string as specified in Clauses 4.2.8 and 4.2.4 of GM/T 0003.1–2012.

A3: compute point $S = [h]P_B$ of the elliptic curve; if S is the infinity point, report error and exit.

A4: compute point $[k]P_B = (x_2, y_2)$ of the elliptic curve, convert the data type of x_2, y_2 to bit string as specified in Clauses 4.2.6 and 4.2.5 of GM/T 0003.1–2012.

A5: compute $t = KDF(x_2||y_2, klen)$. If t is an all-zero bit string, go to A1.

A6: compute $C_2 = M \oplus t$.

A7: compute $C_3 = Hash(x_2||M||y_2)$.

A8: output the ciphertext $C = C_1||C_2||C_3$.

Note: Examples of the process of encryption are described in Annex A.

6.2 Process of encryption algorithm

The process of the encryption algorithm is depicted in Figure 1.

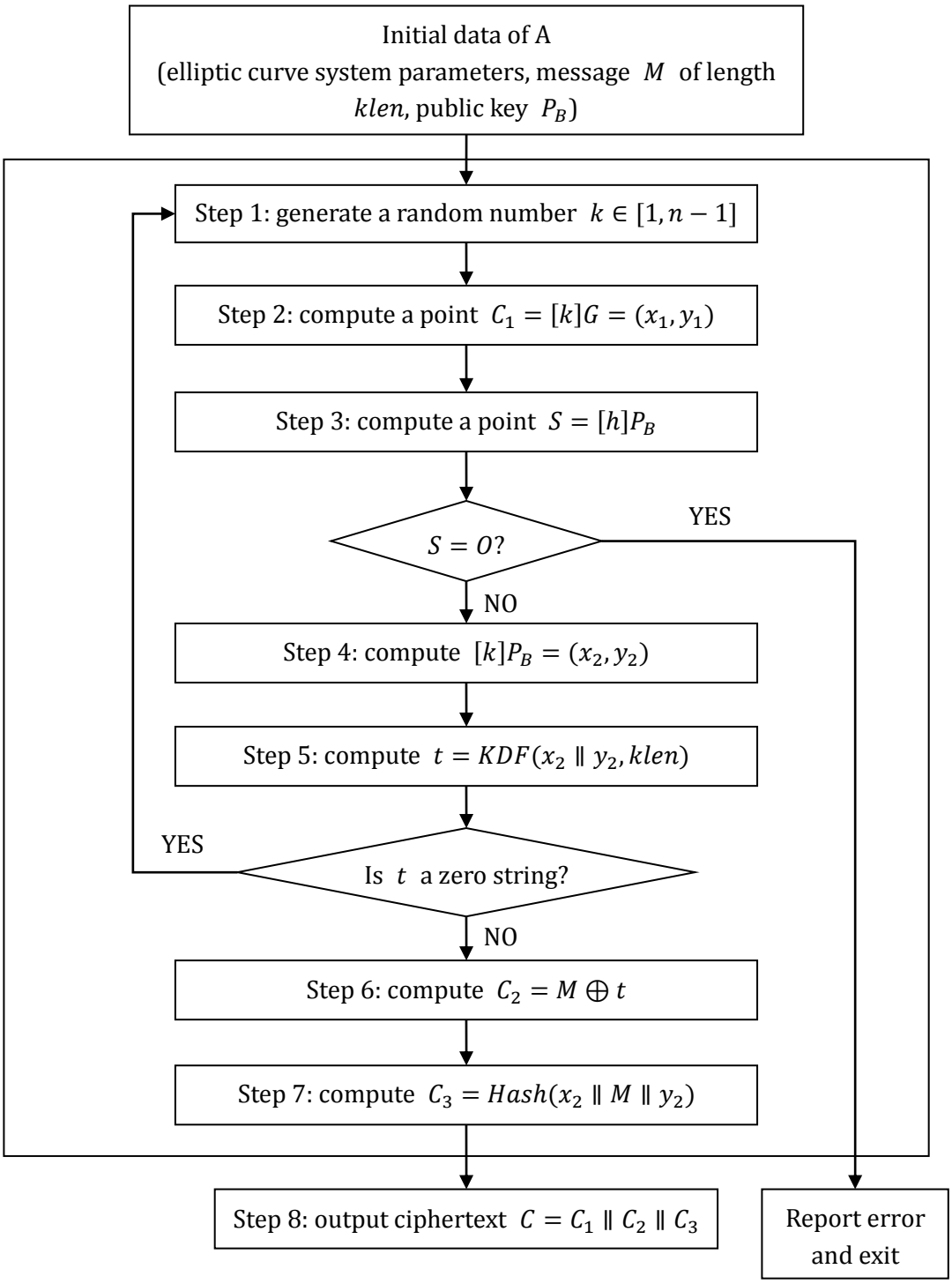


Figure 1: Encryption algorithm process

7 Decryption algorithm and its process

7.1 Decryption algorithm

Suppose $klen$ is the length of C_2 in the ciphertext.

To decrypt the ciphertext $C = C_1 \| C_2 \| C_3$, decryption user B should perform the following procedures:

B1: get C_1 from C and convert the data type of C_1 to the point of the elliptic curve as specified in Clauses 4.2.3 and 4.2.9 of GM/T 0003.1–2012. Then, verify whether C_1 satisfies the elliptic curve equation. If not, output "ERROR" and exit.

B2: compute point $S = [h]C_1$of the elliptic curve. If S is the infinity point, then output "ERROR" and exit.

B3: compute $[d_B]C_1 = (x_2, y_2)$ and convert the data type of x_2, y_2 to bit string as specified in Clauses 4.2.5 and 4.2.4 of GM/T 0003.1–2012.

B4: compute $t = KDF(x_2 \| y_2, klen)$.If t is all-zero bit string, then output "ERROR" and exit.

B5: get C_2 from C and compute $M' = C_2 \oplus t$.

B6: compute $u = Hash(x_2 \| M' \| y_2)$. Get C_3 from C.If $u \neq C_3$, output "ERROR" and exit.

B7: output the plaintext M'.

Note: Examples of the process of key exchange protocol are described in Annex A.

7.2 Process of decryption algorithm

The process of decryption algorithm is depicted in Figure 2.

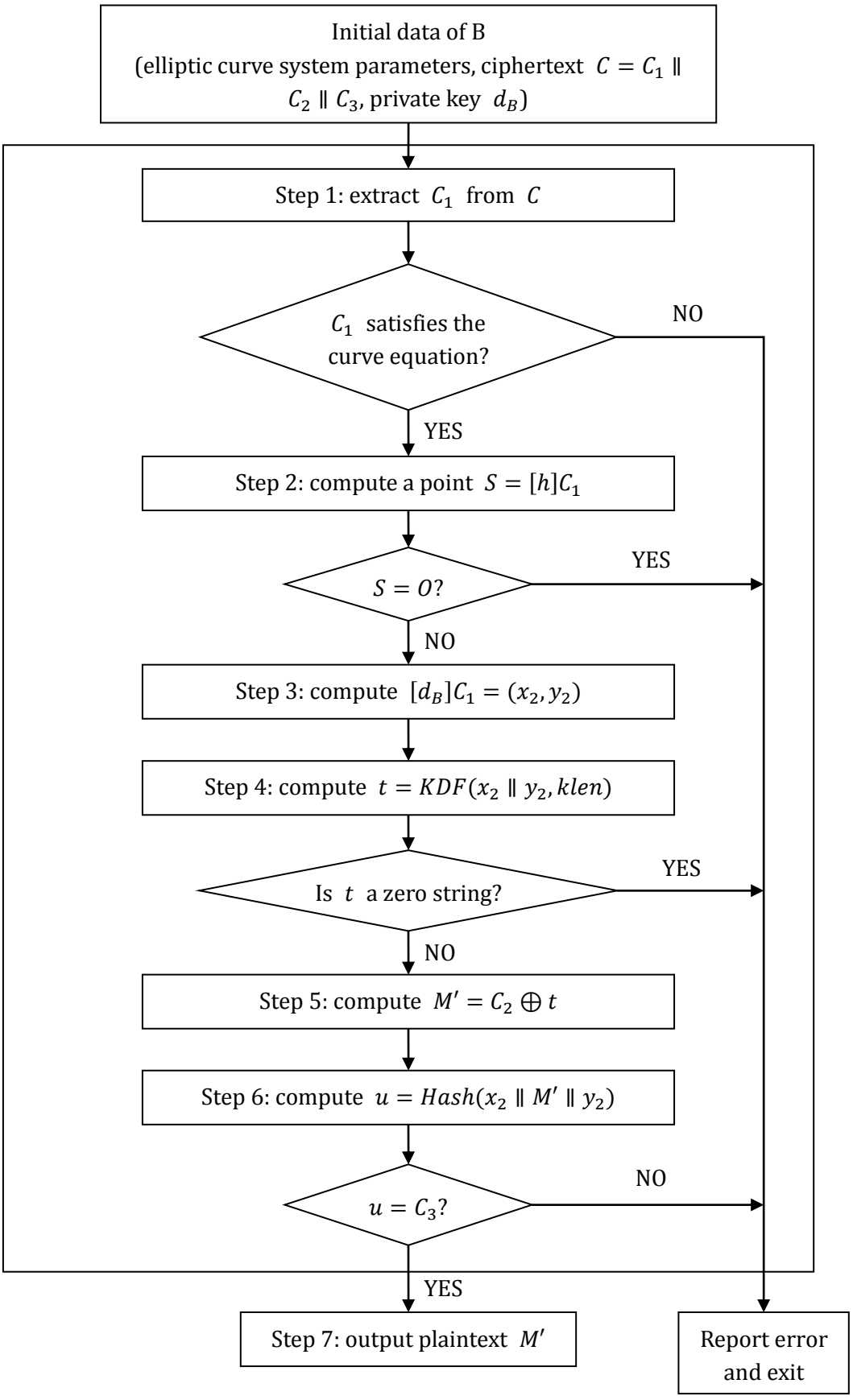


Figure 2: Decryption algorithm process

Annex A
(informative)
Examples of message encryption and decryption

A.1 General requirements

This annex adopts the cryptographic hash function specified in GM/T 0004–2012, SM3 Cryptographic Hash Algorithm, whose input is a bit string of length less than 2^{64}, and output is a hash value of length 256 bits, denoted $H_{256}()$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

In this annex, plaintexts are denoted as ASCII encoding.

A.2 Message encryption and decryption on elliptic curves over F_p

The elliptic curve equation is: $y^2 = x^3 + ax + b$

Example 1: $F_p - 192$

prime p: BDB6F4FE 3E8B1D9E 0DA8C0D4 6F4C318C EFE4AFE3 B6B8551F

coefficient a: BB8E5E8F BC115E13 9FE6A814 FE48AAA6 F0ADA1AA 5DF91985

coefficient b: 1854BEBD C31B21B7 AEFC80AB 0ECD10D5 B1B3308E 6DBF11C1

base point $G = (x_G, y_G)$ whose order is n

coordinate x_G: 4AD5F704 8DE709AD 51236DE6 5E4D4B48 2C836DC6 E4106640

coordinate y_G: 02BB3A02 D4AAADAC AE24817A 4CA3A1B0 14B52704 32DB27D2

order n: BDB6F4FE 3E8B1D9E 0DA8C0D4 0FC96219 5DFAE76F 56564677

message M to be encrypted: encryption standard

hexadecimal form of message M: 656E63 72797074 696F6E20 7374616E 64617264

private key d_B: 58892B80 7074F53F BF67288A 1DFAA1AC 313455FE 60355AFD

public key $P_B = (x_B, y_B)$:

coordinate x_B: 79F0A954 7AC6D100 531508B3 0D30A565 36BCFC81 49F4AF4A

coordinate y_B: AE38F2D8 890838DF 9C19935A 65A8BCC8 994BC792 4672F912

Related values in steps of the encryption algorithm:

generate random number k: 384F3035 3073AEEC E7A16543 30A96204 D37982A3 E15B2CB5

compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 23FC680B 124294DF DF34DBE7 6E0C38D8 83DE4D41 FA0D4CF5

coordinate y_1: 70CF14F2 0DAF0C4D 777F738D 16B16824 D31EEFB9 DE31EE1F

choose the uncompressed form of C_1, convert the point to byte string of form $PC \parallel x_1 \parallel y_1$ where PC is a single byte and $PC = 04$, and denoted still by C_1.

compute point $[k]P_B = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: 57E7B636 23FAE5F0 8CDA468E 872A20AF A03DED41 BF140377

coordinate y_2: 0E040DC8 3AF31A67 991F2B01 EBF9EFD8 881F0A04 93000603

bit length of message M: $klen = 152$

compute $t = KDF(x_2 \parallel y_2, klen)$: 046B04 A9ADF53B 389B9E2A AFB47D90 F4D08978

compute $C_2 = M \oplus t$: 610567 DBD4854F 51F4F00A DCC01CFE 90B1FB1C

compute $C_3 = Hash(x_2 \parallel M \parallel y_2)$:

$x_2 \parallel M \parallel y_2$: 57E7B636 23FAE5F0 8CDA468E 872A20AF A03DED41 BF140377 656E6372 79707469 6F6E2073 74616E64 6172640E 040DC83A F31A6799 1F2B01EB F9EFD888 1F0A0493 000603

C_3 : 6AFB3BCE BD76F82B 252CE5EB 25B57996 86902B8C F2FD8753 6E55EF76 03B09E7C

Output the ciphertext $C = C_1 \parallel C_2 \parallel C_3$:

04 23FC680B 124294DF DF34DBE7 6E0C38D8 83DE4D41 FA0D4CF5 70CF14F2 0DAF0C4D 777F738D 16B16824 D31EEFB9 DE31EE1F 6AFB3BCE BD76F82B 252CE5EB 25B57996 86902B8C F2FD8753 6E55EF76 03B09E7C 610567DB D4854F51 F4F00ADC C01CFE90 B1FB1C

Related values in steps of the decryption algorithm:

compute point $[d_B]C_2 = (x_2, y_2)$:

coordinate x_2: 57E7B636 23FAE5F0 8CDA468E 872A20AF A03DED41 BF140377

coordinate y_2: 0E040DC8 3AF31A67 991F2B01 EBF9EFD8 881F0A04 93000603

compute $t = KDF(x_2 \parallel y_2, klen)$: 046B04 A9ADF53B 389B9E2A AFB47D90 F4D08978

compute $M' = C_2 \oplus t$: 656E63 72797074 696F6E20 7374616E 64617264

comput $u = Hash(x_2 \parallel M' \parallel y_2)$:6AFB3BCE BD76F82B 252CE5EB 25B57996 86902B8C F2FD8753 6E55EF76 03B09E7C

plaintext M': 656E63 72797074 696F6E20 7374616E 64617264, i.e. encryption standard

Example 2: $F_p - 256$

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

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

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

base point $G = (x_G, y_G)$ whose order is n

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

coordinate y_G: 0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2

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

message M to be encrypted: encryption standard

hexadecimal form of message M: 656E63 72797074 696F6E20 7374616E 64617264

private key d_B: 1649AB77 A00637BD 5E2EFE28 3FBF3535 34AA7F7C B89463F2 08DDBC29 20BB0DA0

public key $P_B = (x_B, y_B)$:

coordinate x_B: 435B39CC A8F3B508 C1488AFC 67BE491A 0F7BA07E 581A0E48 49A5CF70 628A7E0A

coordinate y_B: 75DDBA78 F15FEECB 4C7895E2 C1CDF5FE 01DEBB2C DBADF453 99CCF77B BA076A42

Related values in steps of the encryption algorithm:

generate random number k: 4C62EEFD 6ECFC2B9 5B92FD6C 3D957514 8AFA1742 5546D490 18E5388D 49DD7B4F

compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 245C26FB 68B1DDDD B12C4B6B F9F2B6D5 FE60A383 B0D18D1C 4144ABF1 7F6252E7

coordinate y_1: 76CB9264 C2A7E88E 52B19903 FDC47378 F605E368 11F5C074 23A24B84 400F01B8

choose the uncompressed form of C_1, convert the point to byte string of form $PC \parallel x_1 \parallel y_1$ where PC is a single byte and $PC = 04$, and denoted still by C_1.

compute point $[k]P_B = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: 64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE

coordinate y_2: 58D225EC A784AE30 0A81A2D4 8281A828 E1CEDF11 C4219099 84026537 5077BF78

bit length of message M: $klen = 152$

compute $t = KDF(x_2 \parallel y_2, klen)$: 006E30 DAE231B0 71DFAD8A A379E902 64491603

compute $C_2 = M \oplus t$: 650053 A89B41C4 18B0C3AA D00D886C 00286467

compute $C_3 = Hash(x_2 \parallel M \parallel y_2)$:

$x_2 \parallel M \parallel y_2$: 64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE 656E6372 79707469 6F6E2073 74616E64 61726458 D225ECA7 84AE300A 81A2D482 81A828E1 CEDF11C4 21909984 02653750 77BF78

C_3: 9C3D7360 C30156FA B7C80A02 76712DA9 D8094A63 4B766D3A 285E0748 0653426D

output the ciphertext $C = C_1 \parallel C_2 \parallel C_3$:

04 245C26FB 68B1DDDD B12C4B6B F9F2B6D5 FE60A383 B0D18D1C 4144ABF1 7F6252E7 76CB9264 C2A7E88E 52B19903 FDC47378 F605E368 11F5C074 23A24B84 400F01B8 9C3D7360 C30156FA B7C80A02 76712DA9 D8094A63 4B766D3A 285E0748 0653426D 650053A8 9B41C418 B0C3AAD0 0D886C00 286467

Related values in steps of the decryption algorithm:

compute point $[d_B]C_1 = (x_2, y_2)$:

coordinate x_2: 64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE

coordinate y_2: 58D225EC A784AE30 0A81A2D4 8281A828 E1CEDF11 C4219099 84026537 5077BF78

compute $t = KDF(x_2 \parallel y_2, klen)$: 006E30 DAE231B0 71DFAD8A A379E902 64491603

compute $M' = C_2 \oplus t$: 656E63 72797074 696F6E20 7374616E 64617264

comput $u = Hash(x_2 \parallel M' \parallel y_2)$:9C3D7360 C30156FA B7C80A02 76712DA9 D8094A63 4B766D3A 285E0748 0653426D

plaintext M': 656E63 72797074 696F6E20 7374616E 64617264, i.e. encryption standard

A.3 Message encryption and decryption on elliptic curves over F_{2^m}

The elliptic curve equation is: $y^2 + xy = x^3 + ax^2 + b$

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

generator polynomial of base field: $y^{193} + x^{15} + 1$

coefficient a: 0

coefficient b: 00 2FE22037 B624DBEB C4C618E1 3FD998B1 A18E1EE0 D05C46FB

base point $G = (x_G, y_G)$ whose order is n

coordinate x_G: D78D47E8 5C936440 71BC1C21 2CF994E4 D21293AA D8060A84

coordinate y_G: 615B9E98 A31B7B2F DDEEECB7 6B5D8755 86293725 F9D2FC0C

order n: 80000000 00000000 00000000 43E9885C 46BF45D8 C5EBF3A1

message M to be encrypted: encryption standard

hexadecimal form of message M: 656E63 72797074 696F6E20 7374616E 64617264

private key d_B: 6C205C15 89087376 C2FE5FEE E153D4AC 875D643E B8CAF6C5

public key $P_B = (x_B, y_B)$:

coordinate x_B: 00 E788F191 C5591636 FA992CE6 7CDC8D3B 16E4F4D4 6AF267B8

coordinate y_B: 00 BD6E7E5E 4113D790 20ED5A10 287C14B7 A6767C4D 814ADBFD

Related values in steps of the encryption algorithm:

generate random number k: 6E51C537 3D5B4705 DC9B94FA 9BCF30A7 37ED8D69 1E76D9F0

compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 00 95A8B866 7ACF097F 65CE96EB FE53422F CF15876D 16446B8A

coordinate y_1: 01 7A1EC7C9 BAB0DE07 0522311E 75CD31C3 C4D74150 E84E0A95

choose the uncompressed form of C_1, convert the point to byte string of form $PC \parallel x_1 \parallel y_1$ where PC is a single byte and $PC = 04$, and denoted still by C_1.

compute point $[k]P_B = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: 01 C6271B31 F6BE396A 4166C061 6CF4A8AC DA5BEF4D CBF2DD42

coordinate y_2: 01 47AF35DF A1BFE2F1 61521BCF 59BAB835 64868D92 95881735

bit length of message M: $klen = 152$

compute $t = KDF(x_2 \parallel y_2, klen)$: BC5F0D 50F2B2BC F2DC3027 0BAA5249 3B8A67A4

compute $C_2 = M \oplus t$: D9316E 228BC2C8 9BB35E07 78DE3327 5FEB15C0

compute $C_3 = Hash(x_2 \parallel M \parallel y_2)$:

$x_2 \parallel M \parallel y_2$: 01C6271B 31F6BE39 6A4166C0 616CF4A8 ACDA5BEF 4DCBF2DD 42656E63 72797074 696F6E20 7374616E 64617264 0147AF35 DFA1BFE2 F161521B CF59BAB8 3564868D 92958817 35

C_3 : F0A41F6F 48AC723C ECFC4B76 7299A5E2 5C064167 9FBD2D4D 20E9FFD5 B9F0DAB8

output the ciphertext $C = C_1 \parallel C_2 \parallel C_3$:

04 0095A8B8 667ACF09 7F65CE96 EBFE53422FCF 15876D16 446B 8A017A1E C7C9BAB0 DE070522 311E75CD 31C3C4D7 4150E84E 0A95F0A4 1F6F48AC 723CECFC 4B767299 A5E25C06 41679FBD 2D4D20E9 FFD5B9F0 DAB8D931 6E228BC2 C89BB35E 0778DE33 275FEB15 C0

Related values in steps of the decryption algorithm:

compute point $[d_B]C_1 = (x_2, y_2)$:

coordinate x_2: 01 C6271B31 F6BE396A 4166C061 6CF4A8AC DA5BEF4D CBF2DD42

coordinate y_2: 01 47AF35DF A1BFE2F1 61521BCF 59BAB835 64868D92 95881735

compute $t = KDF(x_2 \parallel y_2, klen)$: BC5F0D 50F2B2BC F2DC3027 0BAA5249 3B8A67A4

compute $M' = C_2 \oplus t$: 656E63 72797074 696F6E20 7374616E 64617264

comput $u = Hash(x_2 \parallel M' \parallel y_2)$: F0A41F6F 48AC723C ECFC4B76 7299A5E2 5C064167 9FBD2D4D 20E9FFD5 B9F0DAB8

plaintext M': 656E63 72797074 696F6E20 7374616E 64617264, i.e. encryption standard

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

generator polynomial of base field: $y^{257} + x^{12} + 1$

coefficient a: 0

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

base point $G = (x_G, y_G)$ whose order is n

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

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

order n: 7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

message M to be encrypted: encryption standard

hexadecimal form of message M: 656E63 72797074 696F6E20 7374616E 64617264

private key d_B: 56A270D1 7377AA9A 367CFA82 E46FA526 7713A9B9 1101D077 7B07FCE0 18C757EB

public key $P_B = (x_B, y_B)$:

coordinate x_B: 00 A67941E6 DE8A6180 5F7BCFF0 985BB3BE D986F1C2 97E4D888 0D82B821 C624EE57

coordinate y_B: 01 93ED5A67 07B59087 81B86084 1085F52E EFA7FE32 9A5C8118 43533A87 4D027271

Related values in steps of the encryption algorithm:

generate random number k: 6D3B4971 53E3E925 24E5C122 682DBDC8 705062E2 0B917A5F 8FCDB8EE 4C66663D

compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 01 9D236DDB 305009AD 52C51BB9 32709BD5 34D476FB B7B0DF95 42A8A4D8 90A3F2E1

coordinate y_1: 00 B23B938D C0A94D1D F8F42CF4 5D2D6601 BF638C3D 7DE75A29 F02AFB7E 45E91771

choose the uncompressed form of C_1, convert the point to byte string of form $PC \parallel x_1 \parallel y_1$ where PC is a single byte and $PC = 04$, and denoted still by C_1.

compute point $[k]P_B = (x_2, y_2)$of the elliptic curve:

coordinate x_2: 00 83E628CF 701EE314 1E8873FE 55936ADF 24963F5D C9C64805 66C80F8A 1D8CC51B

coordinate y_2: 01 524C647F 0C0412DE FD468BDA 3AE0E5A8 0FCC8F5C 990FEE11 60292923 2DCD9F36

bit length of message M: $klen = 152$

compute $t = KDF(x_2 \parallel y_2, klen)$: 983BCF 106AB2DC C92F8AEA C6C60BF2 98BB0117

compute $C_2 = M \oplus t$: FD55AC 6213C2A8 A040E4CA B5B26A9C FCDA7373

compute $C_3 = Hash(x_2 \parallel M \parallel y_2)$:

$x_2 \parallel M \parallel y_2$: 0083E628 CF701EE3 141E8873 FE55936A DF24963F 5DC9C648 0566C80F 8A1D8CC5 1B656E63 72797074 696F6E20 7374616E 64617264 01524C64 7F0C0412 DEFD468B DA3AE0E5 A80FCC8F 5C990FEE 11602929 232DCD9F 36

C_3: 73A48625 D3758FA3 7B3EAB80 E9CFCABA 665E3199 EA15A1FA 8189D96F 579125E4

output the ciphertext $C = C_1 \parallel C_2 \parallel C_3$:

04 019D236D DB305009 AD52C51B B932709B D534D476 FBB7B0DF 9542A8A4 D890A3F2 E100B23B 938DC0A9 4D1DF8F4 2CF45D2D 6601BF63 8C3D7DE7 5A29F02A FB7E45E9 177173A4 8625D375 8FA37B3E AB80E9CF CABA665E 3199EA15 A1FA8189 D96F5791 25E4FD55 AC6213C2 A8A040E4 CAB5B26A 9CFCDA73 73

Related values in steps of the decryption algorithm:

compute point $[d_B]C_1 = (x_2, y_2)$:

coordinate x_2: 00 83E628CF 701EE314 1E8873FE 55936ADF 24963F5D C9C64805 66C80F8A 1D8CC51B

coordinate y_2:01 524C647F 0C0412DE FD468BDA 3AE0E5A8 0FCC8F5C 990FEE11 60292923 2DCD9F36

compute $t = KDF(x_2 \parallel y_2, klen)$: 983BCF 106AB2DC C92F8AEA C6C60BF2 98BB0117

compute $M' = C_2 \oplus t$: 656E63 72797074 696F6E20 7374616E 64617264

comput $u = Hash(x_2 \parallel M' \parallel y_2)$: 73A48625 D3758FA3 7B3EAB80 E9CFCABA 665E3199 EA15A1FA 8189D96F 579125E4

plaintext M': 656E63 72797074 696F6E20 7374616E 64617264, i.e. encryption standard

Public Key cryptographic algorithm SM2 based on elliptic curves
Part 5: Parameter definition

Content

Public key cryptographic algorithm SM2 based on elliptic curves

Part 5: Parameter definition

1 Scope

This part of GM/T 0003 specifies the curve parameters of public key cryptographic algorithms SM2 based on elliptic curves, and gives examples of digital signature and verification, key exchange and verification, and message encryption and decryption.

2 Parameter definition

SM2 uses elliptic curves over 256-bit prime fields.

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

Curve parameters:

p=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

a=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

b =28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

n =FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

x_G =32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

y_G =BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

Annex A

(informative)

Example of digital signature and verification

A.1 General requirements

This annex adopts the cryptographic hash function specified in GM/T 0004–2012, SM3 Cryptographic Hash Algorithm, whose input is a bit string of length less than 2^{64}, and output is a hash value of length 256 bits, denoted $H_{256}()$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

In this annex, all messages are denoted as ASCII encoding.

Suppose the ASCII encoding of ID_A is 31323334 35363738 31323334 35363738. $ENTL_A = 0080$.

A.2 SM2 digital signature based on elliptic curves

The equation of elliptic curve is: $y^2 = x^3 + ax + b$

Example: $F_p - 256$

prime p: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

coefficient a: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

coefficient b: 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

base point $G = (x_G, y_G)$ whose order is n

coordinate x_G: 32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

coordinate y_G: BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

order n: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

message to be signed M: message digest

ASCII code of M: 6D65737361676520646967657374

private key d_A: 3945208F 7B2144B1 3F36E38A C6D39F95 88939369 2860B51A 42FB81EF 4DF7C5B8

public key $P_A = (x_A, y_A)$:

coordinate x_A: 09F9DF31 1E5421A1 50DD7D16 1E4BC5C6 72179FAD 1833FC07 6BB08FF3 56F35020

coordinate y_A: CCEA490C E26775A5 2DC6EA71 8CC1AA60 0AED05FB F35E084A 6632F607 2DA9AD13

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: B2E14C5C 79C6DF5B 85F4FE7E D8DB7A26 2B9DA7E0 7CCB0EA9 F4747B8C CDA8A4F3

Intermediate values in the steps of generating signature:

$\bar{M} = Z_A \parallel M$:

B2E14C5C 79C6DF5B 85F4FE7E D8DB7A26 2B9DA7E0 7CCB0EA9 F4747B8C CDA8A4F3 6D657373 61676520 64696765 7374

cryptographic hash value $e = H_{256}(\bar{M})$: F0B43E94 BA45ACCA ACE692ED 534382EB 17E6AB5A 19CE7B31 F4486FDF C0D28640

generate random number k: 59276E27 D506861A 16680F3A D9C02DCC EF3CC1FA 3CDBE4CE 6D54B80D EAC1BC21

compute point: $(x_1, y_1) = [k]G$ of the elliptic curve:

coordinate x_1: 04EBFC71 8E8D1798 62043226 8E77FEB6 415E2EDE 0E073C0F 4F640ECD 2E149A73

coordinate y_1: E858F9D8 1E5430A5 7B36DAAB 8F950A3C 64E6EE6A 63094D99 283AFF76 7E124DF0

compute $r = (e + x_1) \ mod \ n$: F5A03B06 48D2C463 0EEAC513 E1BB81A1 5944DA38 27D5B741 43AC7EAC EEE720B3

$(1 + d_A)^{-1}$: 4DFE9D9C 1F5901D4 E6F58E4E C3D04567 822D2550 F9B88E82 6D1B5B3A B9CD0FE0

Compute $s = ((1 + d_A)^{-1}(k - rd_A)) \ mod \ n$: B1B6AA29 DF212FD8 763182BC 0D421CA1 BB9038FD 1F7F42D4 840B69C4 85BBC1AA

The signature of message M is (r, s):

value r: F5A03B06 48D2C463 0EEAC513 E1BB81A1 5944DA38 27D5B741 43AC7EAC EEE720B3

value s: B1B6AA29 DF212FD8 763182BC 0D421CA1 BB9038FD 1F7F42D4 840B69C4 85BBC1AA

Verify the related values:

cryptographic hash value $e' = H_{256}(\bar{M}')$: F0B43E94 BA45ACCA ACE692ED 534382EB 17E6AB5A 19CE7B31 F4486FDF C0D28640

compute $t = (r' + s')\ mod\ n$: A756E531 27F3F43B 851C47CF EEFD9E43 A2D133CA 258EF4EA 73FBF468 3ACDA13A

compute point $(x_0', y_0') = [s']G$ of the elliptic curve:
coordinate x_0': 2B9CE14E 3C8D1FFC 46D693FA 0B54F2BD C4825A50 6607655D E22894B5 C99D3746

coordinate y_0': 277BFE04 D1E526B4 E1C32726 435761FB CE0997C2 6390919C 4417B3A0 A8639A59compute point $(x_{00}', y_{00}') = [t]P_A$ of the elliptic curve:

coordinate x_{00}': FDAC1EFA A770E463 5885CA1B BFB360A5 84B238FB 2902ECF0 9DDC935F 60BF4F9B

coordinate y_{00}': B89AA926 3D5632F6 EE82222E 4D63198E 78E095C2 4042CBE7 15C23F71 1422D74C

compute point $(x_1', y_1') = [s']G + [t]P_A$ of the elliptic curve

coordinate x_1': 04EBFC71 8E8D1798 62043226 8E77FEB6 415E2EDE 0E073C0F 4F640ECD 2E149A73

coordinate y_1': E858F9D8 1E5430A5 7B36DAAB 8F950A3C 64E6EE6A 63094D99 283AFF76 7E124DF0

compute $R = (e' + x_1')\ mod\ n$: F5A03B06 48D2C463 0EEAC513 E1BB81A1 5944DA38 27D5B741 43AC7EAC EEE720B3

Annex B

(informative)

Examples of key exchange and verification

B.1 General requirements

This annex adopts the cryptographic hash function specified in GM/T 0004–2012, SM3 Cryptographic Hash Algorithm, whose input is a bit string of length less than 2^{64}, and output is a hash value of length 256 bits, denoted $H_{256}()$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

Suppose ASCII encoding of ID_A is 31323334 35363738 31323334 35363738. $ENTL_A = 0080$.

Suppose ASCII encoding of ID_B is: 31323334 35363738 31323334 35363738. $ENTL_B = 0080$.

B.2 SM2 key exchange protocol based on elliptic curves

The equation of elliptic curve is: $y^2 = x^3 + ax + b$

Example : $F_p - 256$

prime p: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

coefficient a: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

coefficient b: 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

cofactor h: 1

base point $G = (x_G, y_G)$ whose order is n

coordinate x_G: 32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

coordinate y_G: BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

order n: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

user A's private key d_A: 81EB26E9 41BB5AF1 6DF11649 5F906952 72AE2CD6 3D6C4AE1 678418BE 48230029

user A's public key $P_A = (x_A, y_A)$:

coordinate x_A: 160E1289 7DF4EDB6 1DD812FE B96748FB D3CCF4FF E26AA6F6 DB9540AF 49C94232

coordinate y_A: 4A7DAD08 BB9A4595 31694BEB 20AA489D 6649975E 1BFCF8C4 741B78B4 B223007F

user B's private key d_B: 78512991 7D45A9EA 5437A593 56B82338 EAADDA6C EB199088 F14AE10D EFA229B5

user B's public key $P_B = (x_B, y_B)$:

coordinate x_B: 6AE848C5 7C53C7B1 B5FA99EB 2286AF07 8BA64C64 591B8B56 6F7357D5 76F16DFB

coordinate y_B: EE489D77 1621A27B 36C5C799 2062E9CD 09A92643 86F3FBEA 54DFF693 05621C4D

hash value $Z_A = H_{256}(ENTL_A \parallel ID_A \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_A \parallel y_A)$

Z_A: 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1

hash value $Z_B = H_{256}(ENTL_B \parallel ID_B \parallel a \parallel b \parallel x_G \parallel y_G \parallel x_B \parallel y_B)$

Z_B: 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7

Related values in step A1-A3 in the key exchange protocol:

generate random number r_A: D4DE1547 4DB74D06 491C440D 305E0124 00990F3E 390C7E87 153C12DB 2EA60BB3

compute point $R_A = [r_A]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 64CED1BD BC99D590 049B434D 0FD73428 CF608A5D B8FE5CE0 7F150269 40BAE40E

coordinate y_1: 376629C7 AB21E7DB 26092249 9DDB118F 07CE8EAA E3E7720A FEF6A5CC 062070C0

Related values in step B1-B9 in the key exchange protocol:

generate random number r_B: 7E071248 14B30948 9125EAED 10111316 4EBF0F34 58C5BD88 335C1F9D 596243D6

compute point $R_B = [r_B]G = (x_2, y_2)$ of the elliptic curve:

coordinate x_2: ACC27688 A6F7B706 098BC91F F3AD1BFF 7DC2802C DB14CCCC DB0A9047 1F9BD707

coordinate y_2: 2FEDAC04 94B2FFC4 D6853876 C79B8F30 1C6573AD 0AA50F39 FC87181E 1A1B46FE

take $\overline{x_2} = 2^{127} + (x_2 \& (2^{127} - 1))$: FDC2802C DB14CCCC DB0A9047 1F9BD707

compute $t_B = (d_B + \overline{x_2} \cdot r_B) \bmod n$: D0429637 F5A6D5D1 E6C54523 5169DF85 23116306 0A654ECB A0F657FD 629E8DD9

take $\overline{x_1} = 2^{127} + (x_1 \& (2^{127} - 1))$: CF608A5D B8FE5CE0 7F150269 40BAE40E

compute the point $[\overline{x_1}]R_A = (x_{A0}, y_{A0})$ of the elliptic curve:

coordinate x_{A0}: 8D62DAF7 DC084E4A 85D32214 68605854 5837BDC2 2D6E9AFE 015828A8 E1094EC2

coordinate y_{A0}: 564DC0FA 639B2967 E65F3448 CA06627E F3FE67C2 1561C5BE BB399552 29A84760

compute point $P_A + [\overline{x_1}]R_A = (x_{A1}, y_{A1})$ of the elliptic curve:

coordinate x_{A1}: 85C40F88 CECA80E3 8172093F C4BA4581 88E7C58A F81CF2AF 454EC431 43E55615

coordinate y_{A1}: 8C152CB0 A131C958 C279DEBE CC6AB739 6A7BC875 FC801BB2 94C284F4 7F65F6ED

compute $V = [h \cdot t_B](P_A + [\overline{x_1}]R_A) = (x_V, y_V)$:

coordinate x_V: C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F

coordinate y_V: 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7

compute $K_B = KDF(x_V \| y_V \| Z_A \| Z_B, klen)$:

$x_V \| y_V \| Z_A \| Z_B$:
C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7

$klen = 128$

shared secret key K_B: 6C893473 54DE2484 C60B4AB1 FDE4C6E5

compute optional term $S_B = Hash(0x02 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:
C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7 64CED1BD BC99D590 049B434D 0FD73428 CF608A5D B8FE5CE0 7F150269 40BAE40E 376629C7 AB21E7DB 26092249 9DDB118F 07CE8EAA E3E7720A FEF6A5CC 062070C0 ACC27688 A6F7B706 098BC91F F3AD1BFF 7DC2802C DB14CCCC DB0A9047 1F9BD707 2FEDAC04 94B2FFC4 D6853876 C79B8F30 1C6573AD 0AA50F39 FC87181E 1A1B46FE

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:
90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

$0x02 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:
02 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

optional term S_B: D3A0FE15 DEE185CE AE907A6B 595CC32A 266ED7B3 367E9983 A896DC32 FA20F8EB

Related values in steps A4-A10 in the key exchange protocol:

take $\overline{x_1} = 2^{127} + (x_1 \& (2^{127} - 1))$: CF608A5D B8FE5CE0 7F150269 40BAE40E

compute $t_A = (d_A + \overline{x_1} \cdot r_A) \bmod n$: 3D68C0C0 6DC40F17 B9DDFE00 93D3C0E4 969ED112 4A187FA8 AD02F81E 3C11CCE6

take $\overline{x_2} = 2^{127} + (x_2 \& (2^{127} - 1))$: FDC2802C DB14CCCC DB0A9047 1F9BD707

compute point $[\overline{x_2}]R_B = (x_{B0}, y_{B0})$ of the elliptic curve:

coordinate x_{B0}: DA68EF84 FE616D92 438BBE69 BCC52DB9 CE5CBEA9 93944CBC 331BA26D 6082E912

coordinate y_{B0}: 4831E862 898B4356 32D8FFA0 1869CD65 645822BD D3B4E9E0 46BCAB85 6F02F110

compute point $P_B + [\overline{x_2}]R_B = (x_{B1}, y_{B1})$ of the elliptic cuve:

coordinate x_{B1}: FE7C111C C3E628E3 FE709DF2 E6E331CD C2A3A30E EA0CDC3C D10C0759 EAB15199

coordinate y_{B1}: 12D6F496 361948C9 EC67E603 DF93C008 86EFAEEA C591C2D5 D16B67F2 FE1AD77E

compute $U = [h \cdot t_A](P_B + [\overline{x_2}]R_B) = (x_U, y_U)$:

coordinate x_U: C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F

coordinate y_U: 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7

compute $K_A = KDF(x_U \parallel y_U \parallel Z_A \parallel Z_B, klen)$:

$x_U \parallel y_U \parallel Z_A \parallel Z_B$:

C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7

$klen = 128$

shared secret key K_A: 6C893473 54DE2484 C60B4AB1 FDE4C6E5

compute optional term $S_1 = Hash(0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$:

C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7 64CED1BD BC99D590 049B434D 0FD73428 CF608A5D B8FE5CE0 7F150269 40BAE40E 376629C7 AB21E7DB 26092249 9DDB118F 07CE8EAA E3E7720A FEF6A5CC 062070C0 ACC27688 A6F7B706 098BC91F F3AD1BFF 7DC2802C DB14CCCC DB0A9047 1F9BD707 2FEDAC04 94B2FFC4 D6853876 C79B8F30 1C6573AD 0AA50F39 FC87181E 1A1B46FE

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

$0x02 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

02 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

optional term S_1: D3A0FE15 DEE185CE AE907A6B 595CC32A 266ED7B3 367E9983 A896DC32 FA20F8EB

compute optional term $S_A = Hash(0x03 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$):

C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7 64CED1BD BC99D590 049B434D 0FD73428 CF608A5D B8FE5CE0 7F150269 40BAE40E 376629C7 AB21E7DB 26092249 9DDB118F 07CE8EAA E3E7720A FEF6A5CC 062070C0 ACC27688 A6F7B706 098BC91F F3AD1BFF 7DC2802C DB14CCCC DB0A9047 1F9BD707 2FEDAC04 94B2FFC4 D6853876 C79B8F30 1C6573AD 0AA50F39 FC87181E 1A1B46FE

$Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

$0x03 \parallel y_U \parallel Hash(x_U \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

optional term S_A: 18C7894B 3816DF16 CF07B05C 5EC0BEF5 D655D58F 779CC1B4 00A4F388 4644DB88

Related values in step B10 in the key exchange protocol:

compute optional term $S_2 = Hash(0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2))$:

$x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2$):

C558B44B EE5301D9 F52B44D9 39BB5958 4D75B903 4DD6A9FC 82687210 9A65739F 3B85A571 79E11E7E 513AA622 991F2CA7 4D1807A0 BD4D4B38 F90987A1 7AC245B1 79C988D6 3229D97E F19FE02C A1056E01 E6A7411E D24694AA 8F834F4A 4AB022F7 64CED1BD BC99D590 049B434D 0FD73428 CF608A5D B8FE5CE0 7F150269 40BAE40E 376629C7 AB21E7DB 26092249 9DDB118F 07CE8EAA E3E7720A FEF6A5CC 062070C0 ACC27688 A6F7B706 098BC91F F3AD1BFF 7DC2802C DB14CCCC DB0A9047 1F9BD707 2FEDAC04 94B2FFC4 D6853876 C79B8F30 1C6573AD 0AA50F39 FC87181E 1A1B46FE

$Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

$0x03 \parallel y_V \parallel Hash(x_V \parallel Z_A \parallel Z_B \parallel x_1 \parallel y_1 \parallel x_2 \parallel y_2)$:

03 3252B35B 191D8AE0 1CD122C0 25204334 C5EACF68 A0CB4854 C6A7D367 ECAD4DE7 90E2A628 E4F57ABD 78339EA3 3F967D11 A154117B EA442F7B 627D4F4D D047B7F6

optional term S_2: 18C7894B 3816DF16 CF07B05C 5EC0BEF5 D655D58F 779CC1B4 00A4F388 4644DB88

Annex C

(informative)

Example of message encryption and decryption

C.1 General requirements

This annex adopts the cryptographic hash function specified in GM/T 0004–2012, SM3 Cryptographic Hash Algorithm, whose input is a bit string of length less than 2^{64}, and output is a hash value of length 256 bits, denoted $H_{256}(\cdot)$.

In this annex, for all values represented in hexadecimal form, the left is the most significant side and the right is the least significant side.

In this annex, plaintexts are denoted as ASCII encoding.

C.2 SM2 message encryption and decryption on elliptic curves

The elliptic curve equation is: $y^2 = x^3 + ax + b$

Example: $F_p - 256$

prime p: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

coefficient a: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

coefficient b: 28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

base point $G = (x_G, y_G)$ whose order is n

coordinate x_G: 32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

coordinate y_G: BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

order n: FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

message M to be encrypted: encryption standard

hexadecimal form of message M: 656E63 72797074 696F6E20 7374616E 64617264

private key d_B: 3945208F 7B2144B1 3F36E38A C6D39F95 88939369 2860B51A 42FB81EF 4DF7C5B8

public key $P_B = (x_B, y_B)$:

coordinate x_B: 09F9DF31 1E5421A1 50DD7D16 1E4BC5C6 72179FAD 1833FC07 6BB08FF3 56F35020

coordinate y_B: CCEA490C E26775A5 2DC6EA71 8CC1AA60 0AED05FB F35E084A 6632F607 2DA9AD13

Related values in steps of the encryption algorithm:

generate random number k: 59276E27 D506861A 16680F3A D9C02DCC EF3CC1FA 3CDBE4CE 6D54B80D EAC1BC21

compute point $C_1 = [k]G = (x_1, y_1)$ of the elliptic curve:

coordinate x_1: 04EBFC71 8E8D1798 62043226 8E77FEB6 415E2EDE 0E073C0F 4F640ECD 2E149A73

coordinate y_1: E858F9D8 1E5430A5 7B36DAAB 8F950A3C 64E6EE6A 63094D99 283AFF76 7E124DF0

choose the uncompressed form of C_1, convert the point to be byte string of form $PC \parallel x_1 \parallel y_1$ where PC is a single byte and $PC = 04$, and denoted still by C_1.

compute point $[k]P_B = (x_2, y_2)$of the elliptic curve:

coordinate x_2: 335E18D7 51E51F04 0E27D468 138B7AB1 DC86AD7F 981D7D41 6222FD6A B3ED230D

coordinate y_2: AB743EBC FB22D64F 7B6AB791 F70658F2 5B48FA93 E54064FD BFBED3F0 BD847AC9

bit length of message M: $klen = 152$

compute $t = KDF(x_2 \parallel y_2, klen)$: 44E60F DBF0BAE8 14376653 74BEF267 49046C9E

compute $C_2 = M \oplus t$: 21886C A989CA9C 7D580873 07CA9309 2D651EFA

compute $C_3 = Hash(x_2 \parallel M \parallel y_2)$:

$x_2 \parallel M \parallel y_2$:

335E18D7 51E51F04 0E27D468 138B7AB1 DC86AD7F 981D7D41 6222FD6A B3ED230D 656E6372 79707469 6F6E2073 74616E64 617264AB 743EBCFB 22D64F7B 6AB791F7 0658F25B 48FA93E 54064FDB FBED3F0B D847AC9

C_3: 59983C18 F809E262 923C53AE C295D303 83B54E39 D609D160 AFCB1908 D0BD8766

output the cipher-text $C = C_1 \parallel C_2 \parallel C_3$:

04EBFC71 8E8D1798 62043226 8E77FEB6 415E2EDE 0E073C0F 4F640ECD 2E149A73 E858F9D8 1E5430A5 7B36DAAB 8F950A3C 64E6EE6A 63094D99 283AFF76 7E124DF0 59983C18 F809E262 923C53AE C295D303 83B54E39 D609D160 AFCB1908 D0BD8766 21886CA9 89CA9C7D 58087307 CA93092D 651EFA

Related values in steps of the decryption algorithm:

compute point $[d_B]C_2 = (x_2, y_2)$:

coordinate x_2: 335E18D7 51E51F040 E27D4681 38B7AB1D C86AD7F9 81D7D416 222FD6AB 3ED230D

coordinate y_2: AB743EBC FB22D64F 7B6AB791 F70658F2 5B48FA93 E54064FD BFBED3F0 BD847AC9

compute $t = KDF(x_2 \parallel y_2, klen)$: 44E60F DBF0BAE8 14376653 74BEF267 49046C9E

compute $M' = C_2 \oplus t$: 656E63 72797074 696F6E20 7374616E 64617264

comput $u = Hash(x_2 \parallel M' \parallel y_2)$:59983C18 F809E262 923C53AE C295D303 83B54E39 D609D160 AFCB1908 D0BD8766

plaintext M': 656E63 72797074 696F6E20 7374616E 64617264, i.e. encryption standard