Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

S. Shen, Ed.
X. Lee, Ed.
Chinese Academy of Science

February 14, 2014

SM2 Digital Signature Algorithm
draft-shen-sm2-ecdsa-02

Abstract

This document discribles a set of public key cryptographic algorithms based on elliptic curves which is invented by Xiaoyun Wang et al. These algorithms and recommended parameters are published by Chinese Commercial Cryptography Administration Office ([SM2 Algorithms] and [SM2 Algorithms Parameters]) for the use of electronic authentication service system. This document gives IETF standard description of the algorithms and parameters in [SM2 Algorithms] and [SM2 Algorithms Parameters].

The document [SM2 Algorithms] published by Chinese Commercial Cryptography Administration Office includes four parts: general introdocution, Digital Signature Algorithm, Key Exchange Protocol and Public Key Encryption Algorithm.

The document [SM2 Algorithms Parameters] gives a set of recommended parameters.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 18, 2014.

Copyright Notice

Table of Contents

1. Introduction

The algorithms and parameters described in this document are published by Chinese Commercial Cryptography Administration Office.the convenience of IETF and IRTF community. This document gives IETF standard description of these algorithms and parameters.

The document [SM2 Algorithms] published by Chinese Commercial Cryptography Administration Office includes four parts: general introdocution, Digital Signature Algorithm, Key Exchange Protocol and Public Key Encryption Algorithm. The content in these four parts are described in section 4, 5, 6, 7.

The document [SM2 Algorithms Parameters] gives a set of recommended parameters. The content in this document is described in Appendix D.

2. Conventions Used in this Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

3. Symbols and Terms

3.1. Symbols

a, b	Elements in finite field Fq and they defines a Elliptic Curve E over Fq
B	The MOV threshold. This is a positive integer B such that taking discrete logarithms over GF (q^B) is judged to be at least as difficult as taking elliptic discrete logarithms over GF (q).
deg(f)	The degree of a polynomial f(x)
E	The elliptic curve defined by a and b over a finite field Fq
E(Fq)	The set of all the rational points of E
#E(Fq)	The number of elements in E(Fq), the degree of elliptic curve E(Fq)
ECDLP	Elliptic Curve Discrete Logarithm Problem
Fp	A prime field with p elements
Fq	A prime field with q elements
F*q	The multiplicative group composed of all non-zero elememnts in Fq
F2^m	The binary field extension with 2^m elements
G	A base point on the elliptic curve E, with prime order
gcd(x;y)	The greatest common devisor of x and y
h	The cofactor h= #E(Fp)/n, where n is the degree of a base point G
LeftRotate( )	The operation of Rotation to left
lmax	The upper limit of the largest prime factor of the cofactor h
m	The extention degree of the field F2^m over the binary field F2

modf(x)	The operation module the polynomial f(x). All the coefficients mod 2 when f(x) is a polynomial over F2.
modn	The operation of modulo n, for example, 23 mod7 = 2
n	The degree of a base point G (n is a prime factor of #E(Fq)
O	The point of infinity (or zero) on the elliptic curre E.
P	A point P on the elliptic curre E which is not O. The coordinates xP and yP satisfies the elliptic curve equation
P1+P2	The summation of the two points P1 and P2 on elliptic curve E
p	A prime number greater than 3
q	The number of elements in the finite field Fq
rmin	The lower limit of the degree n of a base point G
Tr( )	The trace function
xP	The x-coordinate of the point P
yP	The y-coordinate of the point P
x^(-1)	The only y such that x*y=1 (modn), 1 < = y < = n, gcd(x, n)=1
x||y	The concatenation of x and y, where x and y are bit string or byte string
x == y (modn)	x modn = y modn
** y~P	The point compression expression of yP
Zp	The ring of integers modulo p
< G >	The cyclic group generated by base point G
[k]P	The k multiple of a point P over elliptic curve, where k is a positive integer
[x;y]	The set of integers which greater than or equal to x and less than or equal to y
/x\	The smallest integer greater than or equal to x, for example AGBPA[not]/7\=7, /8.3\=9
\x/	The largest integer less than or equal to x, for example AGBPA[not]\7/=7, \8.3/=8
XOR	The exclusive-or operation of two bit strings or byte strings of same length
**********	
A,B	The two users using the public key system
a, b	Elements in finite field Fq and they defines a Elliptic Curve E over Fq
dA	The private key of the user A
E(Fq)	The set of all the rational points of E
e	The hash of message M
e’	The hash of message M’
Fq	A prime field with q elements
G	A base point on the elliptic curve E, with prime order
Hv( )	The hash function with output of legnth v bits
IDA	The identifier of user A
M	The message for signature
MA!a&#65533;	The message for verification
modn	The operation of modulo n, for example, 23 mod7 = 2
n	The degree of base point G (n is a prime factor of #E(Fq))
O	The point of infinity (or zero) on the elliptic curre E
PA	The public key of user A
q	The number of elements in the finite field Fq
x||y	The concatenation of x and y, where x and y are bit string or byte string
ZA	The identifier of user A, part of parameters of elliptic curve and hash value of PA
(r,s)	The sent signature

(r',s')	The received signature
[k]P	The k multiple of a point P over elliptic curve, where k is a positive integer
[x;y]	The set of integers which greater than or equal to x and less than or equal to y
/x\	The smallest integer greater than or equal to x, for example AGBPA[not]/7\=7, /8.3\=9
\x/	The largest integer less than or equal to x, for example AGBPA[not]\7/=7, \8.3/=8
#E(Fq)	The number of elements in E(Fq), the degree of elliptic curve E(Fq)

****

3.2. Terms

The following terms are used in this document.

digital signature
: The metadata over some data. It should provide authentication, integrity protection and non repudiation. [ANSI X9.63-2001]

message
: The bits string of arbitary length. [ISO/IEC 15946-4 3.7]

signed message
: The data composed of a message and its digital signature. [ISO/IEC 15946-4 3.14]

key
: A parameter for cryptographic calculation. It was used for encryption or decryption, shared sectet and verification of digital signature. [ANSI X9.63-2001]

4. General Introdoction to ECC

TBD

5. Digital Signature Algorithm

5.1. Digital Signature System

5.1.1. General Rules

In the digital signtature algorithm, one signer generate digital signature over given data and one verifier verifies the validation of the signature. Each signer ownes one public key and one private key. The private key was used for signing and verifier verfifies the signature using the public key. Before generation of the digital signature, the message M and ZA need to be compressed via a hash function; before the verification of the digital signature, the

message M' and ZA need to be compressed via a hash function.

5.1.2. Parameters of Elliptic Curve System

The parameters of an elliptic curve systme include the size q of a finite field Fq (when q=2^m, also include basis representation and irreducible polynomial); the two elements a and b (in Fq) which defines the elliptic curve equation; the base point G=(xG, yG) (G not euqals O), where xG and yG are ellements in Fq; the degree n of G and other optional parameter such as cofactor h.

5.1.3. Key pairs

The user A's key pair include his private key dA and public key PA=[dA]G=(xA, yA).

5.1.4. Auxilary Functions

5.1.4.1. Introduction

The auxilary functions in the elliptic curve digital signature algorithm in this document include hash algorithm and random number generator.

5.1.4.2. Hash Functions

The sm2 digital signature algorithm requires the hash functions approved by Chinese Commercial Cryptography Administration Office, such as sm3.

5.1.4.3. Random Number Generator

The sm2 digital signature algorithm requires random number generators approved by by Chinese Commercial Cryptography Administration Office.

5.1.4.4. Other User Information

As teh signer, User A has the identifier IDA of length entlenA bits, denote ENTLA as the two bytes transformed from the integer entlenA. In the digital signature algorithms in this document, both signer and verifier need to obtain ZA by calculating the hash value of ZA.

ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)

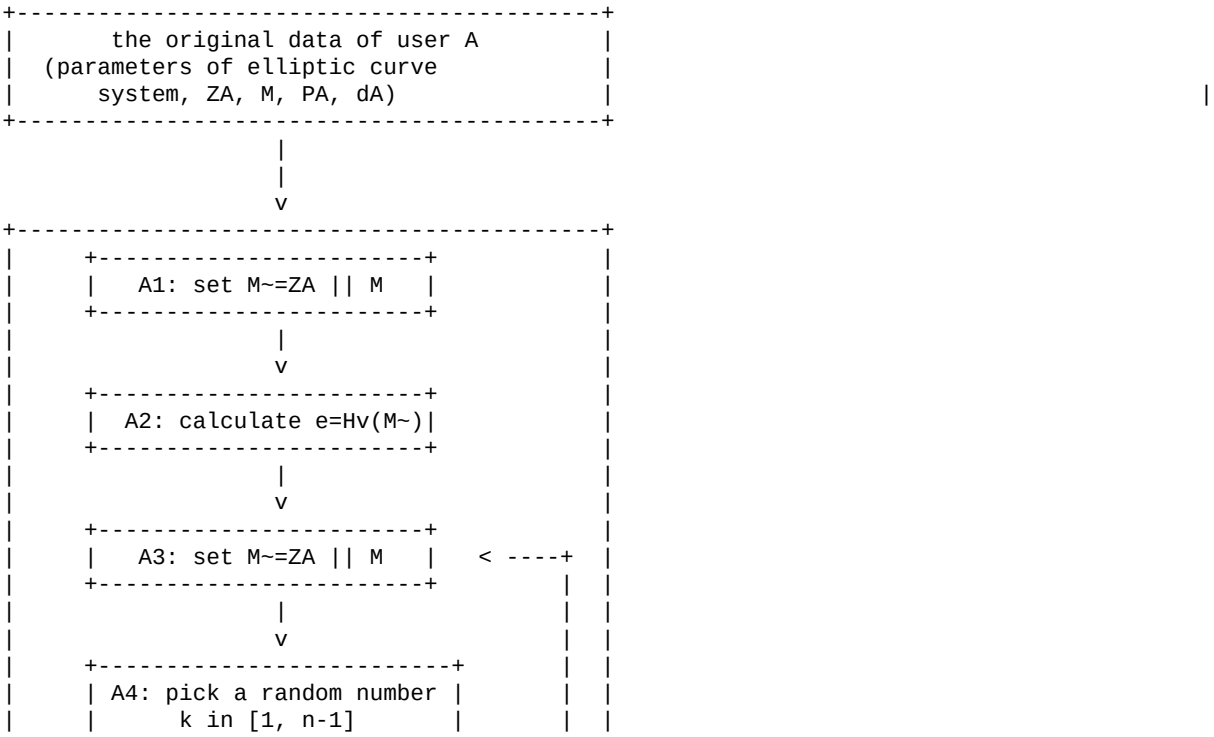
5.2. Generation of Signature

5.2.1. Digital Signature Generation Algorithm

Let M be the message for signing, in order to obtain the signature (r, s), the signer A need to perform the following:

A1: set M~=ZA || M
A2: calculate e=Hv(M~)
A3: pick a random number k in [1, n-1] via a random number generator
A4: calculate the elliptic curve point (x1, y1)=[k]G
A5: calculate r=(e+x1) modn, return to A3 if r=0 or r+k=n
A6: calculate s=((1+dA)^(-1)*(k-r*dA)) modn, return to A3 if s=0
A7: the digital signature of M is (r, s)

5.2.2. Flow Chart of Digital Signature Generation



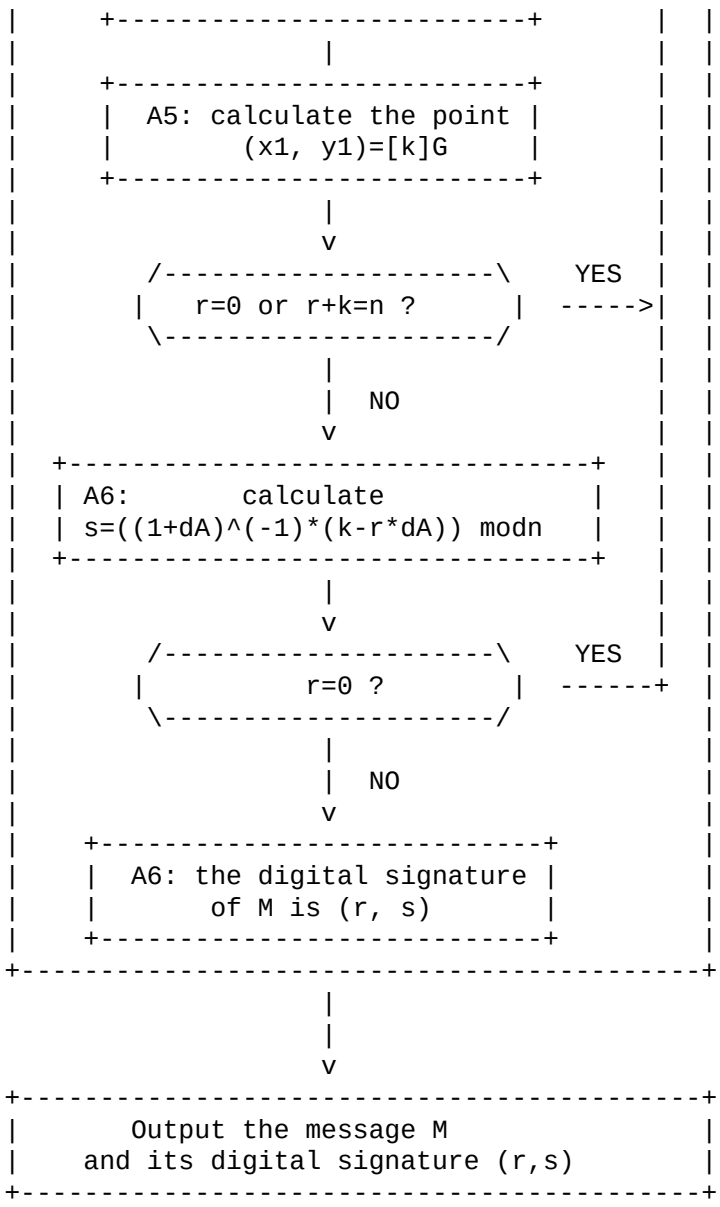


Figure 1: Flow Chart of Digital Signature Generation

5.3. Verification of Signature

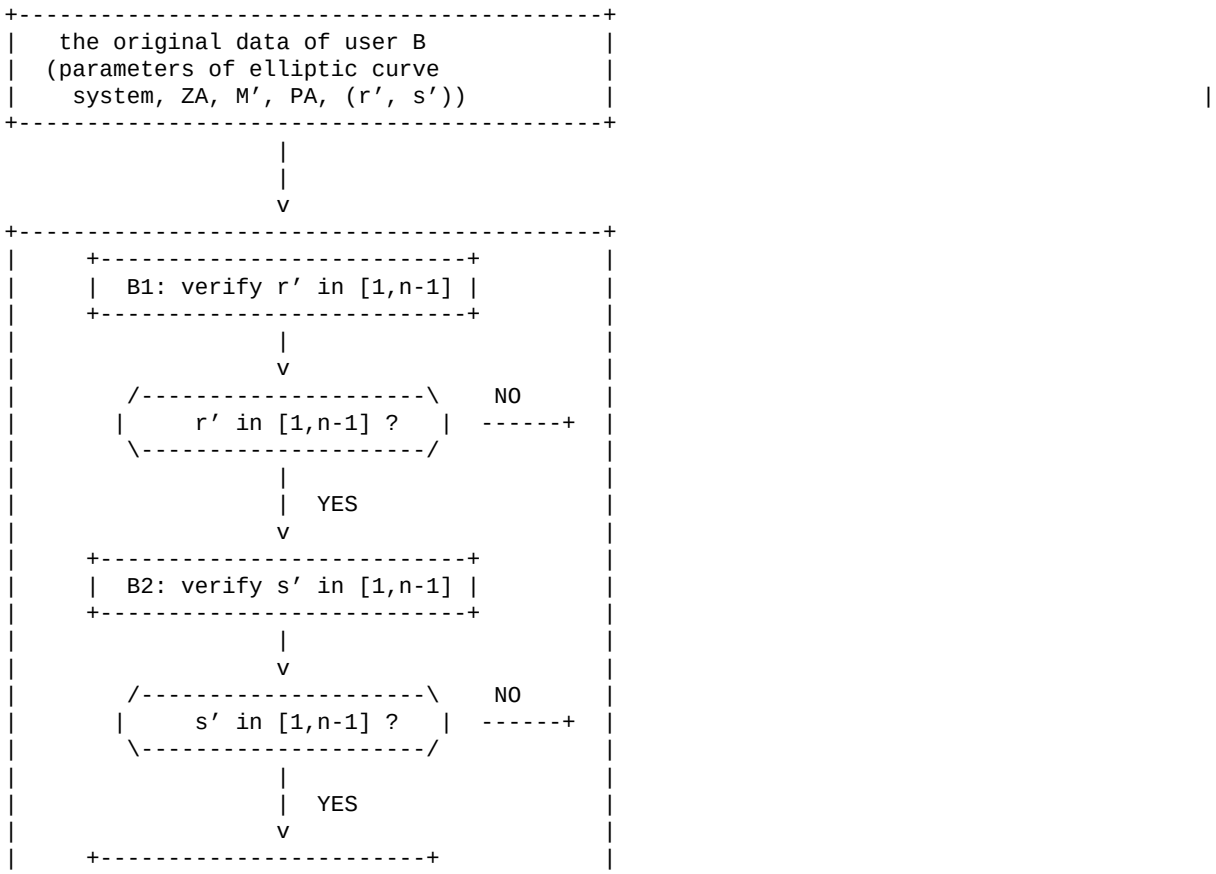
5.3.1. Digital Signagure Vefification Algorithm

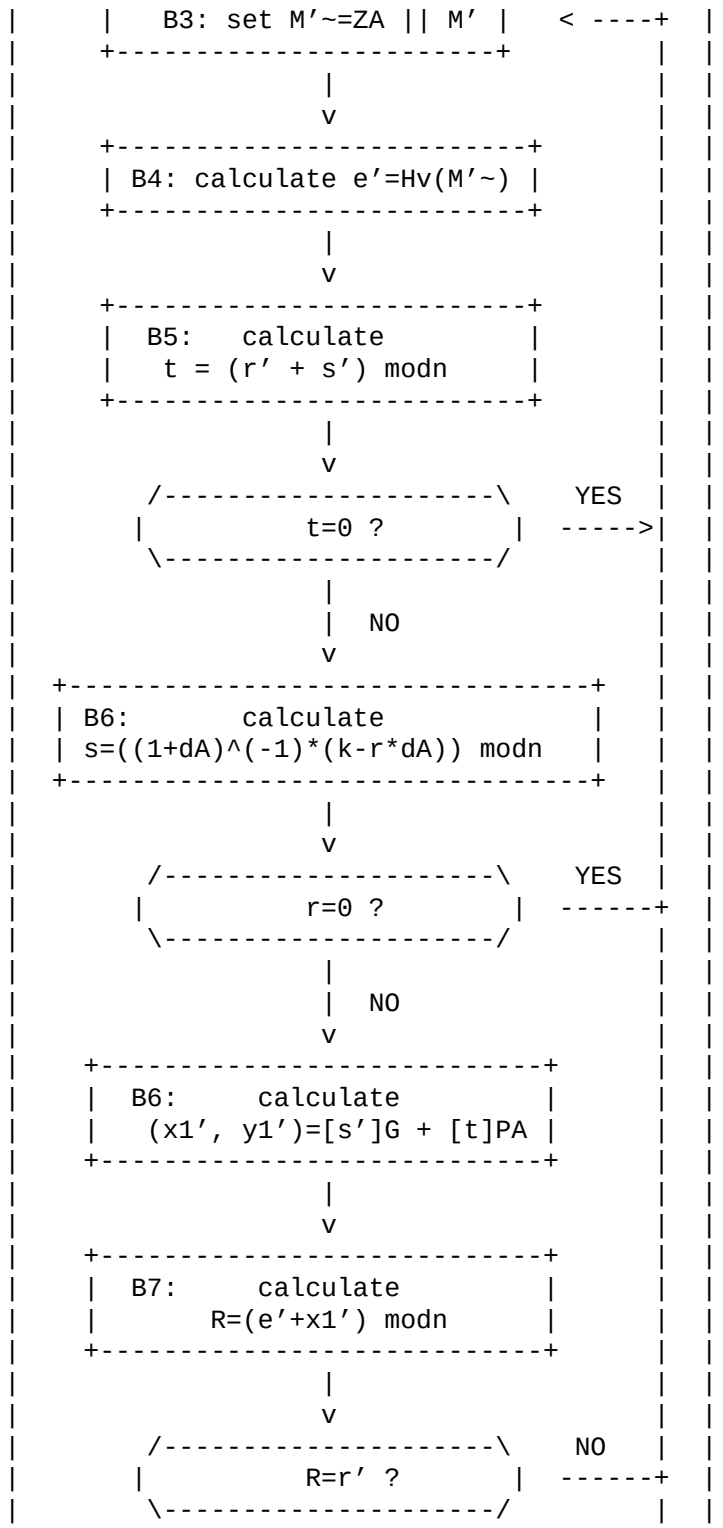
To verfify the received message M' and it's digital signature, the verifier need to perform the following:

B1: verify whether r’ in [1,n-1], verification failed if not
B2: vefify whether s’ in [1,n-1], verification failed if not
B3: set M’~=ZA || M’
B4: calculate e’=Hv(M’~)
B5: calculate t = (r’ + s’) modn, verification failed if t=0
B6: calculate the point (x1’, y1’)=[s’]G + [t]PA
B7: calculate R=(e’+x1’) modn, verfication pass if yes, otherwise failed

Note: The verification will certainly fail if ZA does not correspond to
teh hash value of A.

5.3.2. Flow Chart of Digital Signature Verification




B3: set M'~=ZA || M'
B4: calculate e'=Hv(M'~)
B5: calculate t = (r' + s') modn
t=0 ?
YES
NO
B6: calculate s=((1+dA)^(-1)*(k-r*dA)) modn
r=0 ?
YES
NO
B6: calculate (x1', y1')=[s']G + [t]PA
B7: calculate R=(e'+x1') modn
R=r' ?
NO

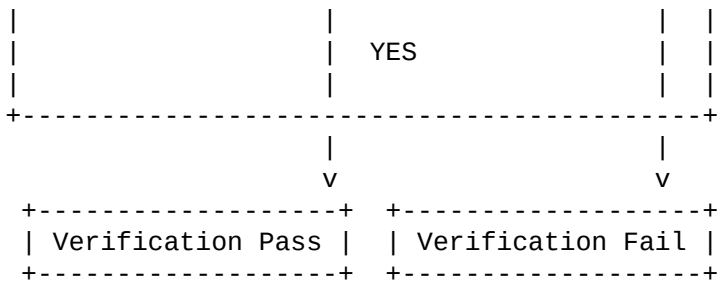


Figure 2: Flow Chart of Digital Signature Verification

6. SM2 Key Exchange Protocol

6.1. Parameters of the Algorithm and Auxiliary Functions

6.1.1. General Rules

In the key exchange protocol, user A and user B use respective private key and opposite public key to get agreement on a secret key only known by themselves through alternate communications. The shared secret key is generally used in a symmetric cryptographic algorithm. The key exchange protocol can be used in key management and key agreement.

6.1.2. Parameters of Elliptic Curve System

The parameters of an elliptic curve systme include the size q of a finite field Fq (when q=2^m, also include basis representation and irreducible polynomial); the two elements a and b (in Fq) which defines the elliptic curve equation; the base point G=(xG, yG) (G not euqals O), where xG and yG are ellements in Fq; the degree n of G and other optional parameter such as cofactor h.

6.1.3. Key pairs

The user A’s key pair include his private key dA and public key PA=[dA]G=(xA, yA). The user B’s key pair include his private key dB and public key PB=[dB]G=(xB, yB).

6.1.4. Auxiliary Functions

6.1.4.1. Introduction

The auxiliary functions in the elliptic curve key exchange protocol in this document include hash functions, key derivation function and random number generator.

6.1.4.2. Hash Function

The sm2 key exchange protocol requires the hash functions approved by Chinese Commercial Cryptography Administration Office, such as sm3.

6.1.4.3. key derivation function

The key derivation function is used for deriving a secret key from a shared secret bit string. In the process of key agreement, the key derivation function acts on a secret bit string shared through key exchange to generate a secret key used for communication or further encryption.
The key derivation function needs to call the hash function.
Let Hv( ) be the hash function whose outputs are hash values of v bits in length.

The key derivation function KDF(Z, klen):
Input: a bit string Z, an integer klen(denoted as the length in bits of secret keys to be obtained, which is supposed to be less than (2^32-1)*v).
Output: a bit string of klen bits in length as the secret key.
a) Initialize a counter of 32 bits, i.e. ct=0x00000001;
b) From i=1 to / klen/v \, do:
   b.1) calculate Ha(i)=Hv(Z || ct);
   b.2) ct++;
c) Let Ha!(/ klen/v \) equal Ha(/ klen/v \) if klen/v is an integer, and let Ha!(/ klen/v \) be the left (klen-(v*\ klen/v /)) bits of Ha(/ klen/v \) if not.
d) let K=Ha(1) || Ha(2) || ... || Ha(/ klen/v \-1) || Ha!(/ klen/v \).

6.1.4.4. Random Number Generator

The sm2 key exchange protocol requires random number generators approved by Chinese Commercial Cryptography Administration Office.

6.1.5. Other User Information

User A has the identifier IDA of length entlenA bits, denote ENTLA as the two bytes transformed from the integer entlenA; User B has the identifier IDB of length entlenB bits, denote ENTLB as the two bytes transformed from the integer entlenB. In the key exchange protocol in this document, both A and B as the participants of key agreement need to obtain ZA and ZB by calculating the hash value of ZA and ZB.

ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)
ZB=H256(ENTLB || IDB || a || b || xG || yG || xB || yB)

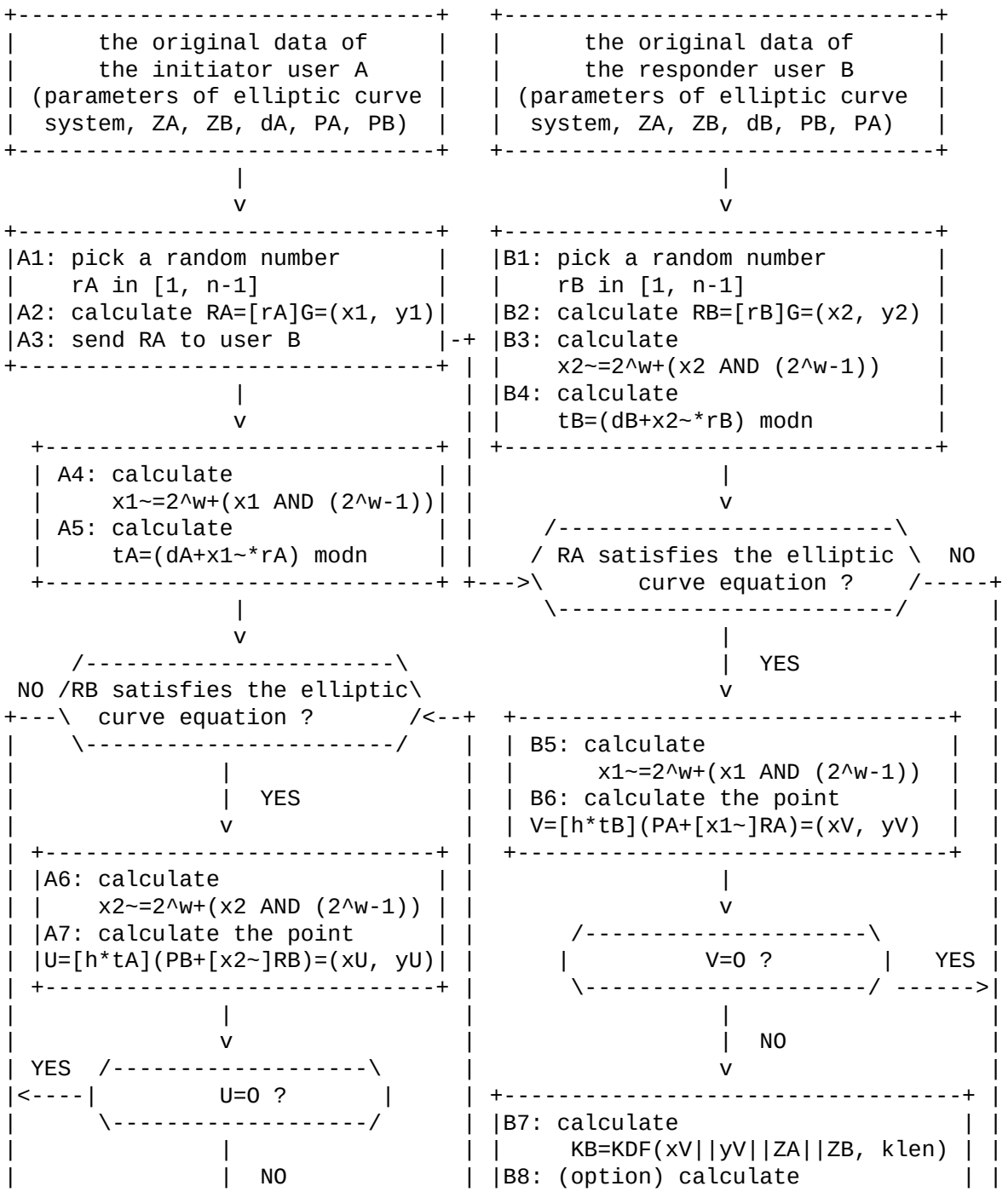
6.2. Key Exchange Protocol and the Flow Chart

6.2.1. Key Exchange Protocol

Let user A be the initiator, user B be the responder and klen be the length in bits of the secret key agreed by user A and user B.
In order to obtain the identical secret key by both user A and user B, they need to perform the following:
Set w=/(/log2(n)\/2)\-1
USER A:
A1: pick a random number rA in [1, n-1] via a random number generator;
A2: calculate the elliptic curve point RA=[rA]G=(x1, y1);
A3: send RA to user B;
USER B:
B1: pick a random number rB in [1, n-1] via a random number generator;
B2: calculate the elliptic curve point RB=[rB]G=(x2, y2);
B3: calculate x2~=2^w+(x2 AND (2^w-1));
B4: calculate tB=(dB+x2~*rB) modn;
B5: verify whether RA satisfies the elliptic curve equation, agreement failed if not; otherwise calculate x1~=2^w+(x1 AND (2^w-1));
B6: calculate the elliptic curve point V=[h*tB](PA+[x1~]RA)=(xV, yV), agreement of B failed if V is the point of infinity;
B7: calculate KB=KDF(xV || yV || ZA || ZB, klen);
B8: (option) calculate SB=Hash(0x02 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2));
B9: (option) send RB, (option SB) to user A;
USER A:
A4: calculate x1~=2^w+(x1 AND (2^w-1));
A5: calculate tA=(dA+x1~*rA) modn;
A6: verify whether RB satisfies the elliptic curve equation, agreement failed if not; otherwise calculate x2~=2^w+(x2 AND (2^w-1));
A7: calculate the elliptic curve point U=[h*tA](PB+[x2~]RB)=(xU, yU), agreement of A failed if U is the point of infinity;
A8: calculate KA=KDF(xU || yU || ZA || ZB, klen);
A9: (option) calculate S1=Hash(0x02 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)), verify whether S1 equals SB, key confirmation from B to A failed if not;
A10: (option) calculate SA=Hash(0x03 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)), send SA to user B;
USER B:
B10: (option) calculate S2=Hash(0x03 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2)), verify whether S2 equals SA, key confirmation from A to B failed if not;

Note: The agreement of the shared secret key will certainly fail if ZA or ZB does not correspond to the hash value of A or B.

6.2.2. Flow Chart of Key Exchange Protocol



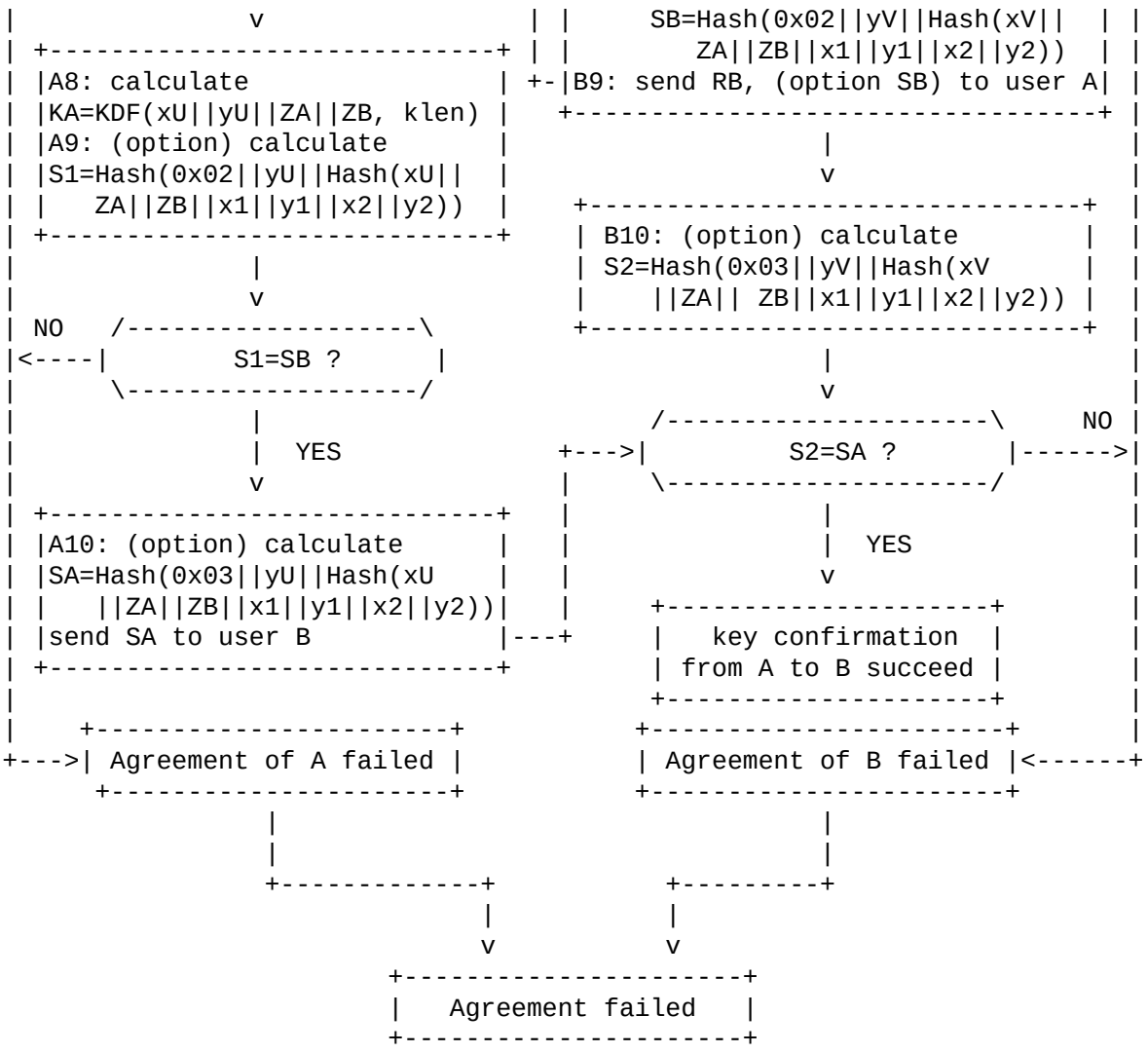


Figure 1: Flow Chart of Key Exchange Protocol

7. SM2 Public Key Encryption Algorithm

7.1. Parameters of the Algorithm and Auxiliary Functions

7.1.1. General Rules

In the public key encryption algorithm, the sender generates ciphertext by encrypting the message with the receiver's public key, and the receiver recovers the original message by decrypting the ciphertext received with his own private key.

7.1.2. Parameters of Elliptic Curve System

The parameters of an elliptic curve systme include the size q of a finite field Fq (when q=2^m, also include basis representation and irreducible polynomial); the two elements a and b (in Fq) which defines the elliptic curve equation; the base point G=(xG, yG) (G not euqals O), where xG and yG are ellements in Fq; the degree n of G and other optional parameter such as cofactor h.

7.1.3. Key pairs

The user B's key pair include his private key dB and public key PB=[dB]G=(xB, yB).

7.1.4. Auxiliary Functions

7.1.4.1. Introduction

The auxiliary functions in the elliptic curve public key encryption algorithm in this document include hash functions, key derivation function and random number generator.

7.1.4.2. Hash Function

The sm2 public key encryption algorithm requires the hash functions approved by Chinese Commercial Cryptography Administration Office, such as sm3.

7.1.4.3. key derivation function

The key derivation function is used for deriving a secret key from a shared secret bit string. In the process of key agreement, the key derivation function acts on a secret bit string shared through key exchange to generate a secret key used for communication or further encryption.
The key derivation function needs to call the hash function.
Let Hv( ) be the hash function whose outputs are hash values of v bits in length.

The key derivation function KDF(Z, klen):
Input: a bit string Z, an integer klen(denoted as the length in bits of secret keys to be obtained, which is supposed to be less than (2^32-1)*v).
Output: a bit string of klen bits in length as the secret key.
a) Initialize a counter of 32 bits, i.e. ct=0x00000001;
b) From i=1 to / klen/v \, do:
b.1) calculate Ha(i)=Hv(Z || ct);
b.2) ct++;
c) Let Ha!(/ klen/v \) equal Ha(/ klen/v \) if klen/v is an integer, and let Ha!(/ klen/v \) be the left (klen-(v*\ klen/v /)) bits of Ha(/ klen/v \) if not.

d) let K=Ha(1) || Ha(2) || ... || Ha(/ klen/v \-1) || Ha!(/ klen/v \).

7.1.4.4. Random Number Generator

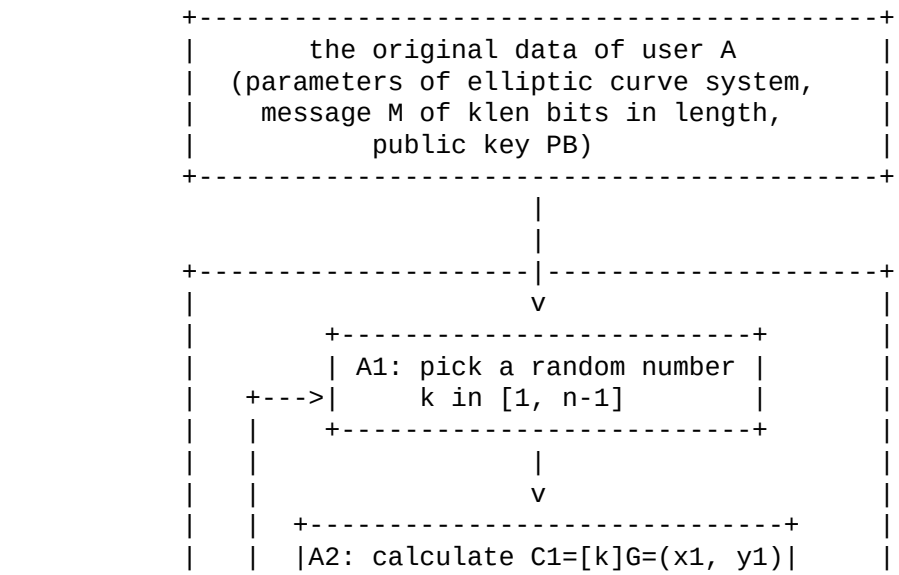
The sm2 public key encryption algorithm requires random number generators approved by Chinese Commercial Cryptography Administration Office.

7.2. Algorithm for Encryption and the Flow Chart

7.2.1. Algorithm for Encryption

Let the bit string M be the message for sending, klen be the length in bits of M.
In order to encrypt M, user A need to perform the following:
A1: pick a random number k in [1, n-1] via a random number generator;
A2: calculate the elliptic curve point C1=[k]G=(x1, y1);
A3: calculate the elliptic curve point S=[h]PB, report error and quit if S is the point of infinity;
A4: calculate the elliptic curve point [k]PB=(x2, y2);
A5: calculate t=KDF(x2 || y2, klen), return to A1 if t is an all zero bit string;
A6: calculate C2=M XOR t;
A7: calculate C3=Hash(x2 || M || y2);
A8: output the ciphertext C=C1 || C2 || C3).

7.2.2. Flow Chart of Algorithm for Encryption



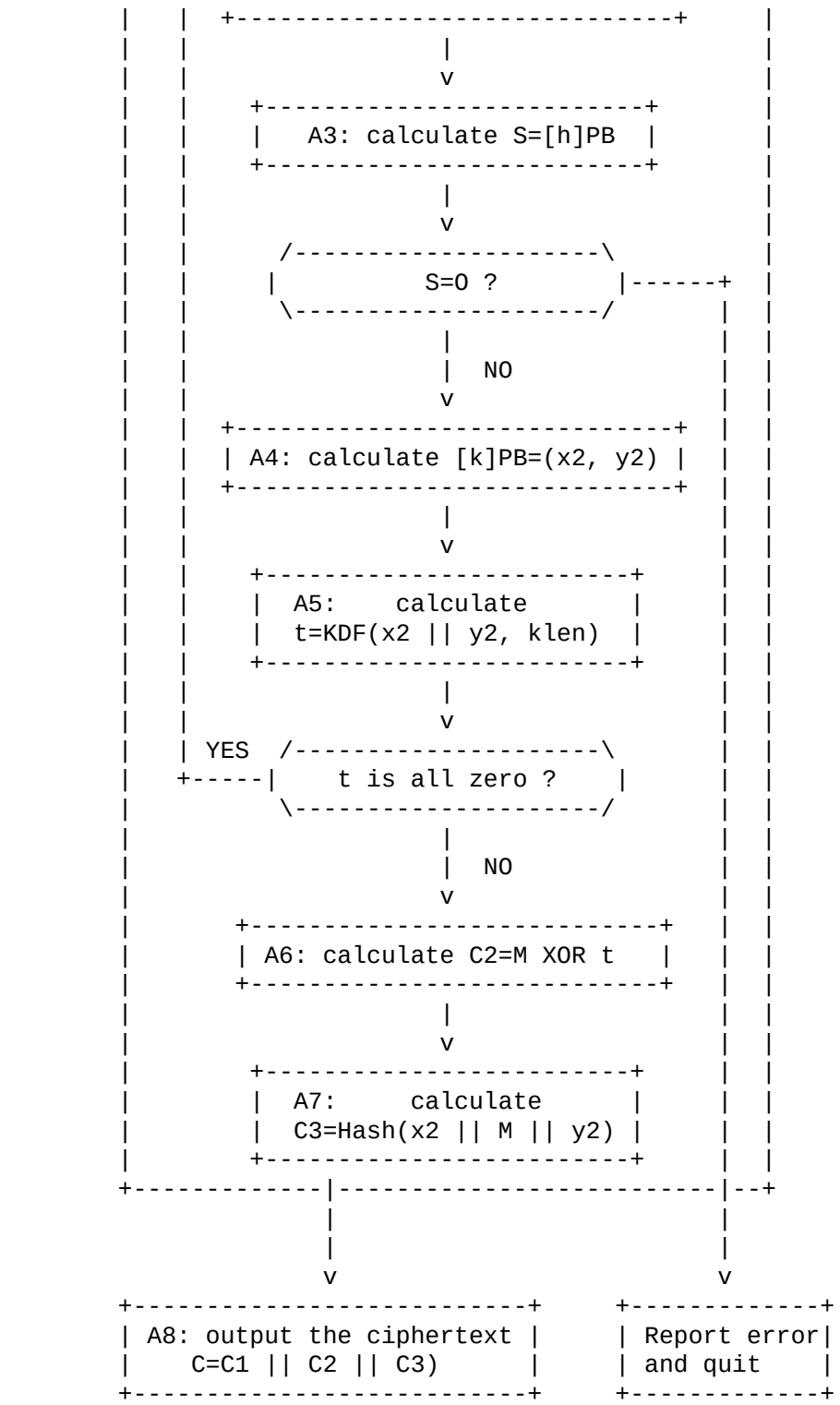

A3: calculate S=[h]PB
S=O ?
NO
A4: calculate [k]PB=(x2, y2)
A5: calculate t=KDF(x2 || y2, klen)
YES
t is all zero ?
NO
A6: calculate C2=M XOR t
A7: calculate C3=Hash(x2 || M || y2)
A8: output the ciphertext C=C1 || C2 || C3)
Report error and quit

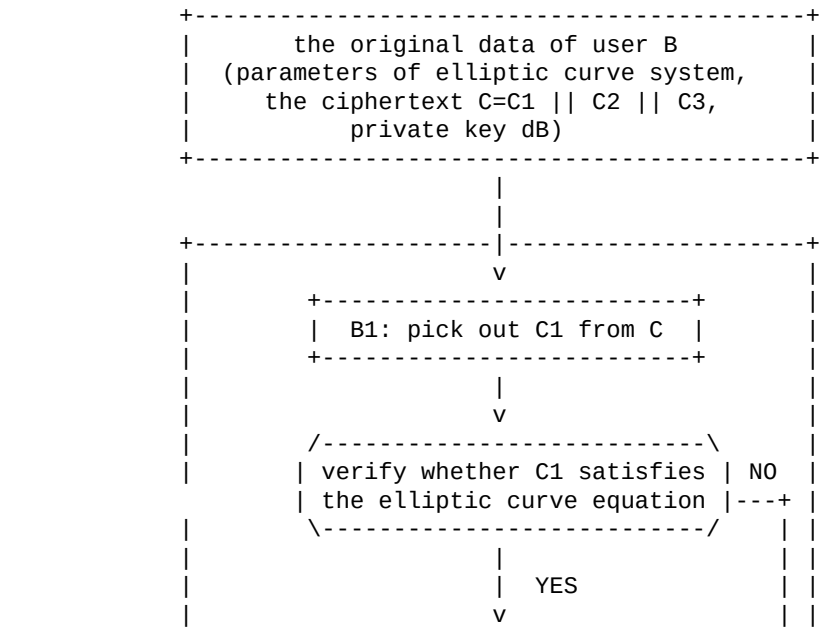
Figure 1: Flow Chart of Algorithm for Encryption

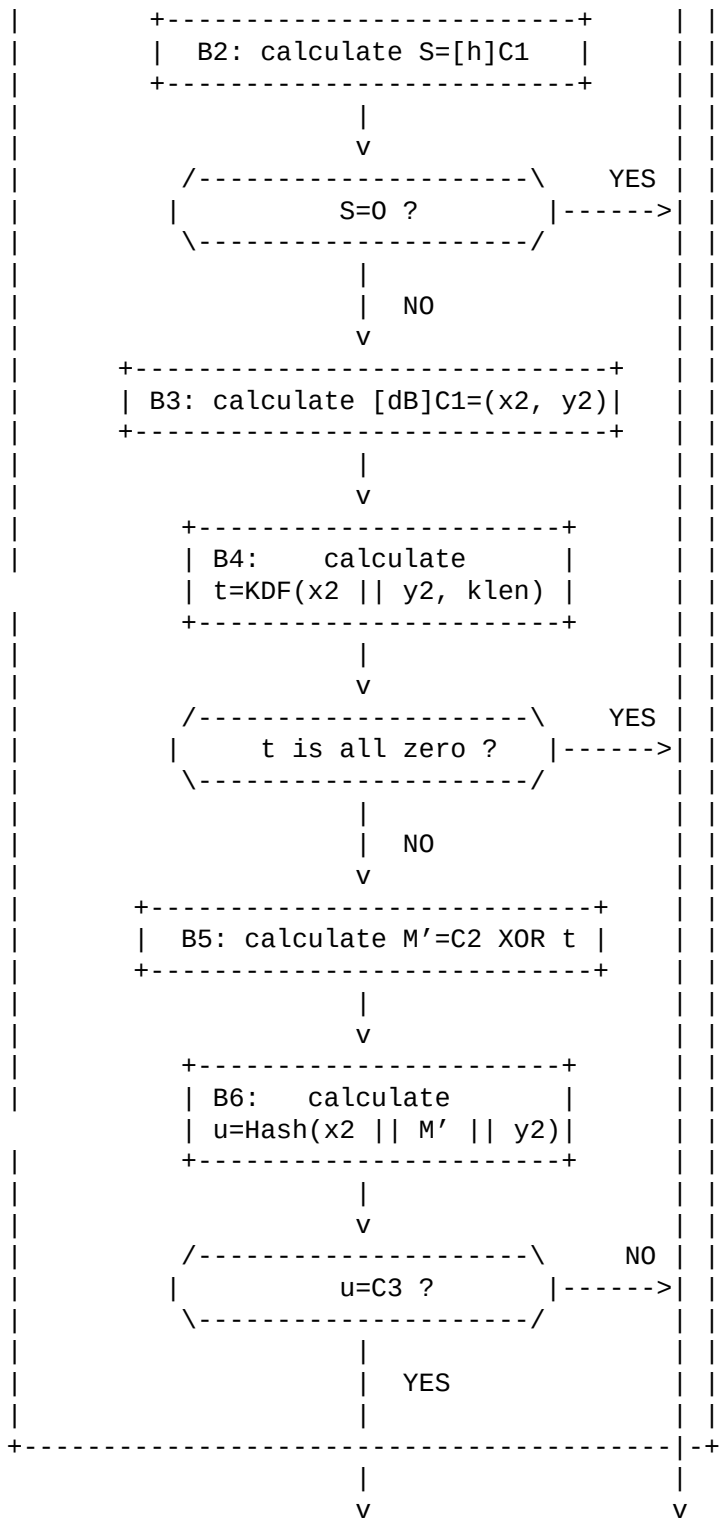
7.3. Algorithm for Decryption and the Flow Chart

7.3.1. Algorithm for Decryption

Let klen be the length in bits of C2 in the ciphertext.
In order to decrypt the ciphertext C=C1 || C2 || C3,
user B need to perform the following:
B1: pick out the bit string C1 from C and transform it into the point on the elliptic curve, verify whether C1 satisfies the elliptic curve equation, report error and quit if not;
B2: calculate the elliptic curve point S=[h]C1, report error and quit if S is the point of infinity;
B3: calculate [dB]C1=(x2, y2);
B4: calculate t=KDF(x2 || y2, klen), report error and quit if t is an all zero bit string;
B5: pick out the bit string C2 from C, calculate M'=C2 XOR t;
B6: calculate u=Hash(x2 || M' || y2), pick out the bit string C3 from C, report error and quit if u doesnot equal C3;
B7: output the plaintext M'.

7.3.2. Flow Chart of Algorithm for Decryption




B2: calculate S=[h]C1
S=O ?
YES
NO
B3: calculate [dB]C1=(x2, y2)
B4: calculate t=KDF(x2 || y2, klen)
t is all zero ?
YES
NO
B5: calculate M'=C2 XOR t
B6: calculate u=Hash(x2 || M' || y2)
u=C3 ?
NO
YES

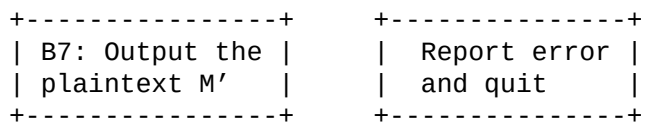


Figure 2: Flow Chart of Algorithm for Decryption

8. Security Considerations

This document gives descriptions of cryptographic algorithms and recommended parameters. There are no known feasible attacks against these algorithms and parameters by the time of publishing this document.

9. References

9.1. Normative References

9.2. Informative References

[SM2 Algorithms] "http://www.oscca.gov.cn/UpFile/2010122214822692.pdf", , December 2010.

[SM2 Algorithms Parameters] "http://www.oscca.gov.cn/UpFile/2010122214836668.pdf", , December 2010.

Appendix A. Examples of Digital Signatures

A.1. General Introduction

This appendix uses the hash algorithm described in draft-shen-sm3-hash-00, which applies on a bit string of length less than 2^54 and output a hash value of size 256, denotes as H256( ).

In this appendix, all the hexadecimal number has high digits on the left and low digits on teh right.

In this appendix, all the messages are in ASCII code.

Let the user A's identity be: ALICE123@YAHOO.COM. Denoted in ASCII code IDA:

414C 49434531 32334059 41484F4F 2E434F4

ENTLA=0090.

A.2. Digital Signature of over E(Fp)

The elliptic curve equationi is:

RFC1341
y^2 = x^3 + ax + b

Example 1: Fp-256
A Prime p:
8542D69E 4C044F18 E8B92435 BF6FF7DE 45728391 5C45517D 722EDB8B 08F1DFC3

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

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

The base point G=(xG,yG)AGBPA[not]whose degree is n:
x-coordinate xG:
421DEBD6 1B62EAB6 746434EB C3CC315E 32220B3B ADD50BDC 4C4E6C14 7FEDD43D
y-coordinate yG:
0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2
degree n:
8542D69E 4C044F18 E8B92435 BF6FF7DD 29772063 0485628D 5AE74EE7 C32E79B7

The message M to be signed:message digest

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

The public key PA=(xA,yA):
x-coordinate xA:
0AE4C779 8AA0F119 471BEE11 825BE462 02BB79E2 A5844495 E97C04FF 4DF2548A
y-coordinate yA:
7C0240F8 8F1CD4E1 6352A73C 17B7F16F 07353E53 A176D684 A9FE0C6B B798E857

Hash value ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)

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

The intermediate value during signing processing:
M~=ZA || M:
F4A38489 E32B45B6 F876E3AC 2168CA39 2362DC8F 23459C1D 1146FC3D BFB7BC9A
6D657373 61676520 64696765 7374
hash value e=H256(M):
B524F552 CD82B8B0 28476E00 5C377FB1 9A87E6FC 682D48BB 5D42E3D9 B9EFFE76

```
random number k:
6CB28D99 385C175C 94F94E93 4817663F C176D925 DD72B727 260DBAAE 1FB2F96F
point (x1,y1)=[k]G:
x-coordinate x1:
110FCDA5 7615705D 5E7B9324 AC4B856D 23E6D918 8B2AE477 59514657 CE25D112
y-coordinate y1:
1C65D68A 4A08601D F24B431E 0CAB4EBE 084772B3 817E8581 1A8510B2 DF7ECA1A
r=(e+x1) modn:
40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1
(1 + dA)^(-1)
79BFCF30 52C80DA7 B939E0C6 914A18CB B2D96D85 55256E83 122743A7 D4F5F956
s = ((1 + dA)^(-1)1 * (k - r * dA)) modn:
6FC6DAC3 2C5D5CF1 0C77DFB2 0F7C2EB6 67A45787 2FB09EC5 6327A67E C7DEEBE7

Digital Signature of the message M: (r,s)
r:
40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1
s:
6FC6DAC3 2C5D5CF1 0C77DFB2 0F7C2EB6 67A45787 2FB09EC5 6327A67E C7DEEBE7

The intermediate value during verification processing:
hash value e' = H256(M'~):
B524F552 CD82B8B0 28476E00 5C377FB1 9A87E6FC 682D48BB 5D42E3D9 B9EFFE76
t=(rA!aa^3A!aS. modn:
2B75F07E D7ECE7CC C1C8986B 991F441A D324D6D6 19FE06DD 63ED32E0 C997C801
point (x0A!aa y0')=[s']G:
x-coordinate x0':
7DEACE5F D121BC38 5A3C6317 249F413D 28C17291 A60DFD83 B835A453 92D22B0A
y-coordinate y0':
2E49D5E5 279E5FA9 1E71FD8F 693A64A3 C4A94611 15A4FC9D 79F34EDC 8BDDEBD0
point (x00', y00')=[t]PA:
x-coordinate x00':
1657FA75 BF2ADCDC 3C1F6CF0 5AB7B45E 04D3ACBE 8E4085CF A669CB25 64F17A9F
y-coordinate y00':
19F0115F 21E16D2F 5C3A485F 8575A128 BBCDDF80 296A62F6 AC2EB842 DD058E50
point (x1', y1')=[s']G + [t]PA:
x-coordinate x1':
110FCDA5 7615705D 5E7B9324 AC4B856D 23E6D918 8B2AE477 59514657 CE25D112
y-coordinate y1':
1C65D68A 4A08601D F24B431E 0CAB4EBE 084772B3 817E8581 1A8510B2 DF7ECA1A
R = (e' + x1') modn:
40F1EC59 F793D9F4 9E09DCEF 49130D41 94F79FB1 EED2CAA5 5BACDB49 C4E755D1
```

A.3. Digital Signature of over E(F2^m)

The elliptic curve equationi is:

```
   y^2 + xy = x^3 + ax + b


Example 1: F2^m -257
The polynomial to generate base field is: x^257 + x^12 + 1

The coefficient a:
0

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

The base point G=(xG,yG)AGBPA[not]whose degree is n:
x-coordinate xG:
00 CDB9CA7F 1E6B0441 F658343F 4B10297C 0EF9B649 1082400A 62E7A748 5735FADD

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

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

The message M to be signed:message digest

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

The public key PA=(xA,yA):
x-coordinate xA:
01 65961645 281A8626 607B917F 657D7E93 82F1EA5C D931F40F 6627F357 542653B2
y-coordinate yA:
01 68652213 0D590FB8 DE635D8F CA715CC6 BF3D05BE F3F75DA5 D5434544 48166612

Hash value ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)
ZA:
26352AF8 2EC19F20 7BBC6F94 74E11E90 CE0F7DDA CE03B27F 801817E8 97A81FD5

The intermediate value during signing processing:
M~=ZA || M:
26352AF8 2EC19F20 7BBC6F94 74E11E90 CE0F7DDA CE03B27F 801817E8 97A81FD5
6D657373 61676520 64696765 7374

hash value e=H256(M~):
AD673CBD A3114171 29A9EAA5 F9AB1AA1 633AD477 18A84DFD 46C17C6F A0AA3B12
random number k:
36CD79FC 8E24B735 7A8A7B4A 46D454C3 97703D64 98158C60 5399B341 ADA186D6
point (x1,y1)=[k]G:
x-coordinate x1:
```

```
00 3FD87D69 47A15F94 25B32EDD 39381ADF D5E71CD4 BB357E3C 6A6E0397 EEA7CD66
y-coordinate y1:
00 80771114 6D73951E 9EB373A6 58214054 B7B56D1D 50B4CD6E B32ED387 A65AA6A2
r=(e+x1) modn:
6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B
(1 + dA)^(-1)
73AF2954 F951A9DF F5B4C8F7 119DAA1C 230C9BAD E60568D0 5BC3F432 1E1F4260
s = ((1 + dA)^(-1)1 * (k - r * dA)) modn:
3124C568 8D95F0A1 0252A9BE D033BEC8 4439DA38 4621B6D6 FAD77F94 B74A9556

Digital Signature of the message M: (r,s)
r:
6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B
s:
3124C568 8D95F0A1 0252A9BE D033BEC8 4439DA38 4621B6D6 FAD77F94 B74A9556

The intermediate value during verification processing:
hash value e' = H256(M'~):
AD673CBD A3114171 29A9EAA5 F9AB1AA1 633AD477 18A84DFD 46C17C6F A0AA3B12
t=(rA!aa^3A!aS. modn:
1E647F8F 784891A6 51AFC342 0316F44A 042D7194 4C91910F 835086C8 2CB07194
point (x0A!aa y0')=[s']G:
x-coordinate x0':
00 252CF6B6 3A044FCE 553EAA77 3E1E9264 44E0DAA1 0E4B8873 89D11552 EA6418F7
y-coordinate y0':
00 776F3C5D B3A0D312 9EAE44E0 21C28667 92E4264B E1BEEBCA 3B8159DC A382653A
point (x00', y00')=[t]PA:
x-coordinate x00':
00 07DA3F04 0EFB9C28 1BE107EC C389F56F E76A680B B5FDEE1D D554DC11 EB477C88
y-coordinate y00':
01 7BA2845D C65945C3 D48926C7 0C953A1A F29CE2E1 9A7EEE6B E0269FB4 803CA68B
point (x1', y1')=[s']G + [t]PA:
x-coordinate x1':
00 3FD87D69 47A15F94 25B32EDD 39381ADF D5E71CD4 BB357E3C 6A6E0397 EEA7CD66
y-coordinate y1':
00 80771114 6D73951E 9EB373A6 58214054 B7B56D1D 50B4CD6E B32ED387 A65AA6A2
R = (e' + x1') modn:
6D3FBA26 EAB2A105 4F5D1983 32E33581 7C8AC453 ED26D339 1CD4439D 825BF25B
```

Appendix B. Examples of Key Exchanges

B.1. General Introduction

This appendix uses the hash algorithm described in draft-shen-sm3-hash-00, which applies on a bit string of length less than 2^64 and output a hash value of size 256, denotes as H256( ).

In this appendix, all the hexadecimal number has high digits on the

left and low digits on the right.

Let the user A's identity be: ALICE123@YAHOO.COM. Denoted in ASCII code IDA:

414C 49434531 32334059 41484F4F 2E434F4D

ENTLA=0090.

Let the user B's identity be: BILL456@YAHOO.COM. Denoted in ASCII code IDB:

42 494C4C34 35364059 41484F4F 2E434F4D

ENTLB=0088.

B.2. Key Exchange Protocal over E(Fp)

The elliptic curve equation is:

y^2 = x^3 + ax + b

Example 1: Fp-256
A Prime p:
8542D69E 4C044F18 E8B92435 BF6FF7DE 45728391 5C45517D 722EDB8B 08F1DFC3

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

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

The cofactor h: 1

The base point G=(xG, yG), whose degree is n:
x-coordinate xG:
421DEBD6 1B62EAB6 746434EB C3CC315E 32220B3B ADD50BDC 4C4E6C14 7FEDD43D
y-coordinate yG:
0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2
degree n:
8542D69E 4C044F18 E8B92435 BF6FF7DD 29772063 0485628D 5AE74EE7 C32E79B7

The private key dA:
6FCBA2EF 9AE0AB90 2BC3BDE3 FF915D44 BA4CC78F 88E2F8E7 F8996D3B 8CCEEDEE

The public key PA=(xA, yA):

```
x-coordinate xA:
3099093B F3C137D8 FCBBCDF4 A2AE50F3 B0F216C3 122D7942 5FE03A45 DBFE1655
y-coordinate yA:
3DF79E8D AC1CF0EC BAA2F2B4 9D51A4B3 87F2EFAF 48233908 6A27A8E0 5BAED98B

The private key dB:
5E35D7D3 F3C54DBA C72E6181 9E730B01 9A84208C A3A35E4C 2E353DFC CB2A3B53

The public key PB=(xB, yB):
x-coordinate xB:
245493D4 46C38D8C C0F11837 4690E7DF 633A8A4B FB3329B5 ECE604B2 B4F37F43
y-coordinate yB:
53C0869F 4B9E1777 3DE68FEC 45E14904 E0DEA45B F6CECF99 18C85EA0 47C60A4C

Hash value ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)

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

Hash value ZB=H256(ENTLB || IDB || a || b || xG || yG || xB || yB)

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

The intermediate value during key exchange processing A1-A3:
random number rA:
83A2C9C8 B96E5AF7 0BD480B4 72409A9A 327257F1 EBB73F5B 073354B2 48668563
point RA=[rA]G=(x1, y1):
x-coordinate x1:
6CB56338 16F4DD56 0B1DEC45 8310CBCC 6856C095 05324A6D 23150C40 8F162BF0
y-coordinate y1:
0D6FCF62 F1036C0A 1B6DACCF 57399223 A65F7D7B F2D9637E 5BBBEB85 7961BF1A

The intermediate value during key exchange processing B1-B9:
random number rB:
33FE2194 0342161C 55619C4A 0C060293 D543C80A F19748CE 176D8347 7DE71C80
point RB=[rB]G=(x2, y2):
x-coordinate x2:
1799B2A2 C7782953 00D9A232 5C686129 B8F2B533 7B3DCF45 14E8BBC1 9D900EE5
y-coordinate y2:
54C9288C 82733EFD F7808AE7 F27D0E73 2F7C73A7 D9AC98B7 D8740A91 D0DB3CF4
x2~=2^127+(x2 AND (2^127-1)):
B8F2B533 7B3DCF45 14E8BBC1 9D900EE5
tB=(dB+x2~*rB) modn:
2B2E11CB F03641FC 3D939262 FC0B652A 70ACAA25 B5369AD3 8B375C02 65490C9F
x1~=2^127+(x1 AND (2^127-1)):
E856C095 05324A6D 23150C40 8F162BF0
point [x1~]RA=(xA0, yA0):
```

```
x-coordinate xA0:
2079015F 1A2A3C13 2B67CA90 75BB2803 1D6F2239 8DD8331E 72529555 204B495B
y-coordinate yA0:
6B3FE6FB 0F5D5664 DCA16128 B5E7FCFD AFA5456C 1E5A914D 1300DB61 F37888ED
point PA+[x1~]RA=(xA1, yA1):
x-coordinate xA1:
1C006A3B FF97C651 B7F70D0D E0FC09D2 3AA2BE7A 8E9FF7DA F32673B4 16349B92
y-coordinate yA1:
5DC74F8A CC114FC6 F1A75CB2 86864F34 7F9B2CF2 9326A270 79B7D37A FC1C145B
point V=[h*tB](PA+[x1~]RA)=(xV, yV):
x-coordinate xV:
47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905
y-coordinate yV:
2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
KB=KDF(xV || yV || ZA || ZB, klen):
xV || yV || ZA || ZB:
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 KB:
55B0AC62 A6B927BA 23703832 C853DED4
option SB=Hash(0x02 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2)):
xV || ZA || ZB || x1 || y1 || x2 || y2:
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(xV || ZA || ZB || x1 || y1 || x2 || y2):
FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
0x02 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2):
02 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
   FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
option SB:
284C8F19 8F141B50 2E81250F 1581C7E9 EEB4CA69 90F9E02D F388B454 71F5BC5C

The intermediate value during key exchange processing A4-A10:
x1~=2^127+(x1 AND (2^127-1)):
E856C095 05324A6D 23150C40 8F162BF0
tA=(dA+x1~*rA) modn:
236CF0C7 A177C65C 7D55E12D 361F7A6C 174A7869 8AC099C0 874AD065 8A4743DC
x2~=2^127+(x2 AND (2^127-1)):
B8F2B533 7B3DCF45 14E8BBC1 9D900EE5
point [x2~]RB=(xB0, yB0):
```

```
x-coordinate xB0:
66864274 6BFC066A 1E731ECF FF51131B DC81CF60 9701CB8C 657B25BF 55B7015D
y-coordinate yB0:
1988A7C6 81CE1B50 9AC69F49 D72AE60E 8B71DB6C E087AF84 99FEEF4C CD523064
point PB+[x2~]RB=(xB1, yB1):
x-coordinate xB1:
7D2B4435 10886AD7 CA3911CF 2019EC07 078AFF11 6E0FC409 A9F75A39 01F306CD
y-coordinate yB1:
331F0C6C 0FE08D40 5FFEDB30 7BC255D6 8198653B DCA68B9C BA100E73 197E5D24
point U=[h*tA](PB+[x2~]RB)=(xU, yU):
x-coordinate xU:
47C82653 4DC2F6F1 FBF28728 DD658F21 E174F481 79ACEF29 00F8B7F5 66E40905
y-coordinate yU:
2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
KA=KDF(xU || yU || ZA || ZB, klen):
xU || yU || ZA || ZB:
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 KA:
55B0AC62 A6B927BA 23703832 C853DED4
option S1=Hash(0x02 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)):
xU || ZA || ZB || x1 || y1 || x2 || y2:
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(xU || ZA || ZB || x1 || y1 || x2 || y2):
FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
0x02 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2):
02 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
   FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
option S1:
284C8F19 8F141B50 2E81250F 1581C7E9 EEB4CA69 90F9E02D F388B454 71F5BC5C
option SA=Hash(0x03 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)):
xU || ZA || ZB || x1 || y1 || x2 || y2:
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(xU || ZA || ZB || x1 || y1 || x2 || y2):
FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
0x03 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2):
03 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
   FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
option SA:
23444DAF 8ED75343 66CB901C 84B3BDBB 63504F40 65C1116C 91A4C006 97E6CF7A

The intermediate value during key exchange processing B10:
option S2=Hash(0x03 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2)):
xV || ZA || ZB || x1 || y1 || x2 || y2:
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(xV || ZA || ZB || x1 || y1 || x2 || y2):
FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
0x03 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2):
03 2AF86EFE 732CF12A D0E09A1F 2556CC65 0D9CCCE3 E249866B BB5C6846 A4C4A295
   FF49D95B D45FCE99 ED54A8AD 7A709110 9F513944 42916BD1 54D1DE43 79D97647
option S2:
23444DAF 8ED75343 66CB901C 84B3BDBB 63504F40 65C1116C 91A4C006 97E6CF7A
```

B.3. Key Exchange Protocal over E(F2^m)

The elliptic curve equation is:

$$y^2 + xy = x^3 + ax + b$$

```
Example 2: F2^m -257
The polynomial to generate base field is: x^257 + x^12 + 1

The coefficient a:
0

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

The cofactor h: 4

The base point G=(xG, yG), whose degree is n:
x-coordinate xG:
00 CDB9CA7F 1E6B0441 F658343F 4B10297C 0EF9B649 1082400A 62E7A748 5735FADD
```

```
y-coordinate yG:
01 3DE74DA6 5951C4D7 6DC89220 D5F7777A 611B1C38 BAE260B1 75951DC8 060C2B3E
degree n:
7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

The private key dA:
4813903D 254F2C20 A94BC570 42384969 54BB5279 F861952E F2C5298E 84D2CEAA

The public key PA=(xA, yA):
x-coordinate xA:
00 8E3BDB2E 11F91933 88F1F901 CCC857BF 49CFC065 FB38B906 9CAAE6D5 AFC3592F
y-coordinate yA:
00 4555122A AC0075F4 2E0A8BBD 2C0665C7 89120DF1 9D77B4E3 EE4712F5 98040415

The private key dB:
08F41BAE 0922F47C 212803FE 681AD52B 9BF28A35 E1CD0EC2 73A2CF81 3E8FD1DC

The public key PB=(xB, yB):
x-coordinate xB:
00 34297DD8 3AB14D5B 393B6712 F32B2F2E 938D4690 B095424B 89DA880C 52D4A7D9
y-coordinate yB:
01 99BBF11A C95A0EA3 4BBD00CA 50B93EC2 4ACB6833 5D20BA5D CFE3B33B DBD2B62D

Hash value ZA=H256(ENTLA || IDA || a || b || xG || yG || xA || yA)

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

Hash value ZB=H256(ENTLB || IDB || a || b || xG || yG || xB || yB)

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

The intermediate value during key exchange processing A1-A3:
random number rA:
54A3D667 3FF3A6BD 6B02EBB1 64C2A3AF 6D4A4906 229D9BFC E68CC366 A2E64BA4
point RA=[rA]G=(x1, y1):
x-coordinate x1:
01 81076543 ED19058C 38B313D7 39921D46 B80094D9 61A13673 D4A5CF8C 7159E304
y-coordinate y1:
01 D8CFFF7C A27A01A2 E88C1867 3748FDE9 A74C1F9B 45646ECA 0997293C 15C34DD8

The intermediate value during key exchange processing B1-B9:
random number rB:
1F219333 87BEF781 D0A8F7FD 708C5AE0 A56EE3F4 23DBC2FE 5BDF6F06 8C53F7AD
point RB=[rB]G=(x2, y2):
x-coordinate x2:
00 2A4832B4 DCD399BA AB3FFFE7 DD6CE6ED 68CC43FF A5F2623B 9BD04E46 8D322A2A
```

```
y-coordinate y2:
00 16599BB5 2ED9EAFA D01CFA45 3CF3052E D60184D2 EECFD42B 52DB7411 0B984C23
x2~=2^127+(x2 AND (2^127-1)):
E8CC43FF A5F2623B 9BD04E46 8D322A2A
tB=(dB+x2~*rB) modn:
3D51D331 14A453A0 5791DB63 5B45F8DB C54686D7 E2212D49 E4A717C6 B10DEDB0
h*tB modn:
75474CC4 52914E81 5E476D8D 6D17E36F 5882EE67 A1CDBC26 FE4122B0 B741A0A3
x1~=2^127+(x1 AND (2^127-1)):
B80094D9 61A13673 D4A5CF8C 7159E304
point [x1~]RA=(xA0, yA0):
x-coordinate xA0:
01 98AB5F14 349B6A46 F77FBFCB DDBFCD34 320DC1F4 C546D13C 3A9F0E83 0C39B579
y-coordinate yA0:
00 BFB49224 ACCE2E51 04CD4519 C0CBE3AD 0C19BF11 805BE108 59069AA6 9317A2B7
point PA+[x1~]RA=(xA1, yA1):
x-coordinate xA1:
00 24A92F64 66A37C5C 12A2C68D 58BFB0F0 32F2B976 60957CB0 5E63F961 F160FE57
y-coordinate yA1:
00 F74A4F17 DC560A55 FDE0F1AB 168BCBF7 6502E240 BA2D6BD6 BE6E5D79 16B288FC
point V=[h*tB](PA+[x1~]RA)=(xV, yV):
x-coordinate xV:
00 DADD0874 06221D65 7BC3FA79 FF329BB0 22E9CB7D DFCFCCFE 277BE8CD 4AE9B954
y-coordinate yV:
01 F0464B1E 81684E5E D6EF281B 55624EF4 6CAA3B2D 37484372 D91610B6 98252CC9
KB=KDF(xV || yV || ZA || ZB, klen):
xV || yV || ZA || ZB:
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 KB:
4E587E5C 66634F22 D973A7D9 8BF8BE23
option SB=Hash(0x02 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2)):
xV || ZA || ZB || x1 || y1 || x2 || y2:
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(xV || ZA || ZB || x1 || y1 || x2 || y2):
E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C
0x02 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2):
```

```
02 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C
   C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C
option SB:
4EB47D28 AD3906D6 244D01E0 F6AEC73B 0B51DE15 74C13798 184E4833 DBAE295A

The intermediate value during key exchange processing A4-A10:
x1~=2^127+(x1 AND (2^127-1)):
B80094D9 61A13673 D4A5CF8C 7159E304
tA=(dA+x1~*rA) modn:
18A1C649 B94044DF 16DC8634 993F1A4A EE3F6426 DFE14AC1 3644306A A5A94187
h*tA modn:
62871926 E501137C 5B7218D2 64FC692B B8FD909B 7F852B04 D910C1AA 96A5061C
x2~=2^127+(x2 AND (2^127-1)):
E8CC43FF A5F2623B 9BD04E46 8D322A2A
point [x2~]RB=(xB0, yB0):
x-coordinate xB0:
01 0AA3BAC9 7786B629 22F93414 57AC64F7 2552AA15 D9321677 A10C7021 33B16735
y-coordinate yB0:
00 C10837F4 8F53C46B 714BCFBF AA1AD627 11FCB03C 0C25B366 BF176A2D C7B8E62E
point PB+[x2~]RB=(xB1, yB1):
x-coordinate xB1:
00 C7A446E1 98DB4278 60C3BB50 ED2197DE B8161973 9141CA61 03745035 9FAD9A99
y-coordinate yB1:
00 602E5A42 17427EAB C5E3917D E81BFFA1 D806591A F949DD7C 97EF90FD 4CF0A42D
point U=[h*tA](PB+[x2~]RB)=(xU, yU):
x-coordinate xU:
00 DADD0874 06221D65 7BC3FA79 FF329BB0 22E9CB7D DFCFCCFE 277BE8CD 4AE9B954
y-coordinate yU:
01 F0464B1E 81684E5E D6EF281B 55624EF4 6CAA3B2D 37484372 D91610B6 98252CC9
KA=KDF(xU || yU || ZA || ZB, klen):
xU || yU || ZA || ZB:
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 KA:
4E587E5C 66634F22 D973A7D9 8BF8BE23
option S1=Hash(0x02 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)):
xU || ZA || ZB || x1 || y1 || x2 || y2:
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(xU || ZA || ZB || x1 || y1 || x2 || y2):
E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C
0x02 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2):
02 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C
   C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C
option S1:
4EB47D28 AD3906D6 244D01E0 F6AEC73B 0B51DE15 74C13798 184E4833 DBAE295A
option SA=Hash(0x03 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2)):
xU || ZA || ZB || x1 || y1 || x2 || y2:
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(xU || ZA || ZB || x1 || y1 || x2 || y2):
E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C
0x03 || yU || Hash(xU || ZA || ZB || x1 || y1 || x2 || y2):
03 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C
   C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C
option SA:
588AA670 64F24DC2 7CCAA1FA B7E27DFF 811D500A D7EF2FB8 F69DDF48 CC0FECB7

The intermediate value during key exchange processing B10:
option S2=Hash(0x03 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2)):
xV || ZA || ZB || x1 || y1 || x2 || y2:
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(xV || ZA || ZB || x1 || y1 || x2 || y2):
E05FE287 B73B0CE6 639524CD 86694311 562914F4 F6A34241 01D885F8 8B05369C
0x03 || yV || Hash(xV || ZA || ZB || x1 || y1 || x2 || y2):
03 01F0464B 1E81684E 5ED6EF28 1B55624E F46CAA3B 2D374843 72D91610 B698252C
   C9E05FE2 87B73B0C E6639524 CD866943 11562914 F4F6A342 4101D885 F88B0536 9C
option S2:
588AA670 64F24DC2 7CCAA1FA B7E27DFF 811D500A D7EF2FB8 F69DDF48 CC0FECB7
```

Appendix C. Example of Public Key Encryption

C.1. General Introduction

This appendix uses the hash algorithm described in draft-shen-sm3-hash-00, which applies on a bit string of length less than 2^64 and output a hash value of size 256, denotes as H256( ).

In this appendix, all the hexadecimal number has high digits on the left and low digits on the right.

In this appendix, all the plaintexts are in ASCII code.

C.2. Encryption and Decryption over E(Fp)

The elliptic curve equation is:

y^2 = x^3 + ax + b

Example 1: Fp-256
A Prime p:
8542D69E 4C044F18 E8B92435 BF6FF7DE 45728391 5C45517D 722EDB8B 08F1DFC3

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

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

The base point G=(xG,yG), whose degree is n:
x-coordinate xG:
421DEBD6 1B62EAB6 746434EB C3CC315E 32220B3B ADD50BDC 4C4E6C14 7FEDD43D
y-coordinate yG:
0680512B CBB42C07 D47349D2 153B70C4 E5D7FDFC BFA36EA1 A85841B9 E46E09A2
degree n:
8542D69E 4C044F18 E8B92435 BF6FF7DD 29772063 0485628D 5AE74EE7 C32E79B7

The message M to be encrypted: encryption standard
M denoted in hexadecimal:
656E63 72797074 696F6E20 7374616E 64617264

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

The public key PB=(xB,yB):
x-coordinate xB:

435B39CC A8F3B508 C1488AFC 67BE491A 0F7BA07E 581A0E48 49A5CF70 628A7E0A
y-coordinate yB:
75DDBA78 F15FEECB 4C7895E2 C1CDF5FE 01DEBB2C DBADF453 99CCF77B BA076A42

The intermediate value during encrypting processing:
random number k:
4C62EEFD 6ECFC2B9 5B92FD6C 3D957514 8AFA1742 5546D490 18E5388D 49DD7B4F
point C1=[k]G=(x1,y1):
x-coordinate x1:
245C26FB 68B1DDDD B12C4B6B F9F2B6D5 FE60A383 B0D18D1C 4144ABF1 7F6252E7
y-coordinate y1:
76CB9264 C2A7E88E 52B19903 FDC47378 F605E368 11F5C074 23A24B84 400F01B8
The point C1 here is uncompressed and can be transformed into a byte string
PC || x1 || x2, where PC is the byte 04. The byte string is still denoted
as C1.
point [k]PB=(x2,y2):
x-coordinate x2:
64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE
y-coordinate y2:
58D225EC A784AE30 0A81A2D4 8281A828 E1CEDF11 C4219099 84026537 5077BF78
the length of message M: klen=152
t=KDF(x2 || y2,klen):
006E30 DAE231B0 71DFAD8A A379E902 64491603
C2=M XOR t:
650053 A89B41C4 18B0C3AA D00D886C 00286467
C3=Hash(x2 || M || y2):
x2 || M || y2:
64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE
656E6372 79707469 6F6E2073 74616E64 61726458 D225ECA7 84AE300A 81A2D482
81A828E1 CEDF11C4 21909984 02653750 77BF78
C3:
9C3D7360 C30156FA B7C80A02 76712DA9 D8094A63 4B766D3A 285E0748 0653426D
ciphertext C=C1 || C2 || C3:
04245C26 FB68B1DD DDB12C4B 6BF9F2B6 D5FE60A3 83B0D18D 1C4144AB F17F6252
E776CB92 64C2A7E8 8E52B199 03FDC473 78F605E3 6811F5C0 7423A24B 84400F01
B8650053 A89B41C4 18B0C3AA D00D886C 00286467 9C3D7360 C30156FA B7C80A02
76712DA9 D8094A63 4B766D3A 285E0748 0653426D

The intermediate value during decrypting processing:
point [dB]C1=(x2,y2):
x-coordinate x2:
64D20D27 D0632957 F8028C1E 024F6B02 EDF23102 A566C932 AE8BD613 A8E865FE
y-coordinate y2:
58D225EC A784AE30 0A81A2D4 8281A828 E1CEDF11 C4219099 84026537 5077BF78
t=KDF(x2 || y2,klen):
006E30 DAE231B0 71DFAD8A A379E902 64491603
M'=C2 XOR t:
656E63 72797074 696F6E20 7374616E 64617264

u=Hash(x2 || M' || y2):
9C3D7360 C30156FA B7C80A02 76712DA9 D8094A63 4B766D3A 285E0748 0653426D
plaintext M':
656E63 72797074 696F6E20 7374616E 64617264
M': encryption standard

C.3. Encryption and Decryption over E(F2^m)

The elliptic curve equation is:

y^2 + xy = x^3 + ax + b

Example 2: F2^m -257
The polynomial to generate base field is: x^257 + x^12 + 1

The coefficient a:
0

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

The base point G=(xG,yG), whose degree is n:
x-coordinate xG:
00 CDB9CA7F 1E6B0441 F658343F 4B10297C 0EF9B649 1082400A 62E7A748 5735FADD
y-coordinate yG:
01 3DE74DA6 5951C4D7 6DC89220 D5F7777A 611B1C38 BAE260B1 75951DC8 060C2B3E
degree n:
7FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF BC972CF7 E6B6F900 945B3C6A 0CF6161D

The message M to be encrypted: encryption standard
M denoted in hexadecimal:
656E63 72797074 696F6E20 7374616E 64617264

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

The public key PB=(xB,yB):
x-coordinate xB:
00 A67941E6 DE8A6180 5F7BCFF0 985BB3BE D986F1C2 97E4D888 0D82B821 C624EE57
y-coordinate yB:
01 93ED5A67 07B59087 81B86084 1085F52E EFA7FE32 9A5C8118 43533A87 4D027271

The intermediate value during encrypting processing:
random number k:
6D3B4971 53E3E925 24E5C122 682DBDC8 705062E2 0B917A5F 8FCDB8EE 4C66663D
point C1=[k]G=(x1,y1):

```
x-coordinate x1:
01 9D236DDB 305009AD 52C51BB9 32709BD5 34D476FB B7B0DF95 42A8A4D8 90A3F2E1
y-coordinate y1:
00 B23B938D C0A94D1D F8F42CF4 5D2D6601 BF638C3D 7DE75A29 F02AFB7E 45E91771
The point C1 here is uncompressed and can be transformed into a byte string
PC || x1 || x2, where PC is the byte 04. The byte string is still denoted
as C1.
point [k]PB=(x2,y2):
x-coordinate x2:
00 83E628CF 701EE314 1E8873FE 55936ADF 24963F5D C9C64805 66C80F8A 1D8CC51B
y-coordinate y2:
01 524C647F 0C0412DE FD468BDA 3AE0E5A8 0FCC8F5C 990FEE11 60292923 2DCD9F36
the length of message M: klen=152
t=KDF(x2 || y2,klen):
983BCF 106AB2DC C92F8AEA C6C60BF2 98BB0117
C2=M XOR t:
FD55AC 6213C2A8 A040E4CA B5B26A9C FCDA7373 FCDA7373
C3=Hash(x2 || M || y2):
x2 || M || y2:
0083E628 CF701EE3 141E8873 FE55936A DF24963F 5DC9C648 0566C80F 8A1D8CC5
1B656E63 72797074 696F6E20 7374616E 64617264 01524C64 7F0C0412 DEFD468B
DA3AE0E5 A80FCC8F 5C990FEE 11602929 232DCD9F 36
C3:
73A48625 D3758FA3 7B3EAB80 E9CFCABA 665E3199 EA15A1FA 8189D96F 579125E4
ciphertext C=C1 || C2 || C3:
04019D23 6DDB3050 09AD52C5 1BB93270 9BD534D4 76FBB7B0 DF9542A8 A4D890A3
F2E100B2 3B938DC0 A94D1DF8 F42CF45D 2D6601BF 638C3D7D E75A29F0 2AFB7E45
E91771FD 55AC6213 C2A8A040 E4CAB5B2 6A9CFCDA 737373A4 8625D375 8FA37B3E
AB80E9CF CABA665E 3199EA15 A1FA8189 D96F5791 25E4

The intermediate value during decrypting processing:
point [dB]C1=(x2,y2):
x-coordinate x2:
00 83E628CF 701EE314 1E8873FE 55936ADF 24963F5D C9C64805 66C80F8A 1D8CC51B
y-coordinate y2:
01 524C647F 0C0412DE FD468BDA 3AE0E5A8 0FCC8F5C 990FEE11 60292923 2DCD9F36
t=KDF(x2 || y2,klen):
983BCF 106AB2DC C92F8AEA C6C60BF2 98BB0117
M'=C2 XOR t:
656E63 72797074 696F6E20 7374616E 64617264
u=Hash(x2 || M' || y2):
73A48625 D3758FA3 7B3EAB80 E9CFCABA 665E3199 EA15A1FA 8189D96F 579125E4
plaintext M':
656E63 72797074 696F6E20 7374616E 64617264
M': encryption standard
```

Appendix D. Recommended Parameters

A elliptic curve on a prime field of 256 bits is recommended:

y^2 = x^3 + ax + b

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
Gx=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7
Gy=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

Authors' Addresses

Sean Shen (editor)
Chinese Academy of Science
No.4 South 4th Zhongguancun Street
Beijing, 100190
China

Phone: +86 10-58813038
EMail: shenshuo@cnnic.cn

Xiaodong Lee (editor)
Chinese Academy of Science
No.4 South 4th Zhongguancun Street
Beijing, 100190
China

Phone: +86 10-58813038
EMail: xl@cnnic.cn