GM/T 0044.4

SM9 Identity-based Cryptographic Algorithms

Part 4: Key Encapsulation Mechanism and Public Key Encryption Algorithm

Cryptography Standardization Technical Committee of China

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

Contents

Foreword

GM/T 0044 “SM9 Identity-based Cryptographic Algorithms” consists of 5 parts:

- Part 1: General
- Part 2: Digital Signature Algorithm
- Part 3: Key Exchange Protocol
- Part 4: Key Encapsulation Mechanism and Public Key Encryption Algorithm
- Part 5: Parameter Definition

This document is the fourth part of GM/T 0044.

1 Scope

This part specifies an identity-based key encapsulation mechanism, and a public key encryption and decryption algorithm built upon pairings from elliptic curves and specifies the corresponding processes. The key encapsulation mechanism can be used to encapsulate a secret key to a specific entity. The public key encryption and decryption algorithms are identity-based asymmetric cryptographic algorithms, which allow the sender to encrypt the message using the identity of a receiver, and only the receiver can decrypt the encrypted message using its corresponding private key.

This part applies to the key encapsulation and the encryption and decryption of a message.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes the requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

GM/T 0004, SM3 Cryptographic Hash Algorithm

GM/T 0002, SM4 Block Cipher Algorithm

GM/T 0044.1–2016, SM9 Identity-based Cryptographic Algorithms — Part 1: General

GM/T 0044.3–2016, SM9 Identity-based Cryptographic Algorithms — Part 3: Key Exchange Protocol

GM/Z 4001, Cryptographic Terminology

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1 secret key

key shared by both the sender and the receiver in a cryptographic system, unknown to any third party

3.2 message

bit string of finite length

3.3 key derivation function

function that generates one or more shared private keys from shared secrets and other parameters known to both entities

3.4 encryption master key

topmost key in the key hierarchy of an identity–based cryptographic system, composed of the encryption master private key and the encryption master public key. The encryption master public key is publicly available while the encryption master private key is kept secret by the KGC. The KGC generates the user's encryption private key by using the encryption master private key and the user's identity. In an identity–based cryptographic system, the encryption master private key is usually generated by the KGC using random number generator while the encryption master public key is generated with the encryption master private key and the system parameters

3.5 identity

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

3.6 key generation center (KGC)

trusted authority responsible for the selection of system parameters, generation of the encryption master keys, and generation of users' encryption private keys (in this part)

4 Symbols

The following symbols apply to this part.

A, B: two users A and B using the identity-based cryptographic system

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

cid : curve identifier that indicates the type of elliptic curve, denoted by one byte, where 0x10 represents an ordinary curve (the non-supersingular curve) over F_p (the prime number $p > 2^{191}$), 0x11 represents a supersingular curve over F_p , and 0x12 represents an ordinary curve and its twisted curve over F_p

$Dec(\ )$: block cipher decryption algorithm

de_B: encryption private key of the user B

$Enc(\ )$: block cipher encryption algorithm

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

eid: bilinear pairing identifier to distinguish the type of the bilinear pairing e, denoted by one byte, where 0x01 represents the Tate pairing, 0x02 represents the Weil pairing, 0x03 represents the Ate pairing, and 0x04 represents the R-Ate pairing

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

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

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

g^u: g to the power of u, where g is an element in the multiplicative group $\mathbb{G}_T$ and u is a positive integer, that is $g^u = \underbrace{g \cdot g \cdot \cdots \cdot g}_{Multiply\ u\ times}$

$H_v(\ )$: a cryptographic hash function

$H_1(\ ), H_2(\ )$: cryptographic functions derived from the cryptographic hash function

hid: identifier of the encryption private key generating function, denoted by one byte, selected and made public by the KGC

ID_B: the identity of user B that uniquely determines the public key of B

$KDF(\ )$: the key derivation function

M: the message to be encrypted

M': the message obtained by decryption

$MAC(\ )$: the message authentication code function

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

P_{pub-e}: the encryption master public key

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

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

ke: the encryption master private key

$\langle P \rangle$: the cyclic group generated by the element P

$[u]P$: the u multiple of the element P in the additive groups $\mathbb{G}_1$ or $\mathbb{G}_2$

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

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

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

β: the twisted curve parameter

5 Algorithm parameters and auxiliary functions

5.1 Overview

The key is a crucial parameter in the control of cryptographic transformations in modern cryptography, and the security of cryptographic output greatly depends on the security of the key. The key encapsulation mechanism enables a user to generate and encrypt a secret key to a target user, such as only the target user can decrypt the secret key, which can be used further as a basis for session keys.

This part specifies an identity-based key encapsulation mechanism realized with elliptic curve pairings. The decapsulating user holds an identity and the corresponding private key, which is generated by the KGC using the master private key and the user's identity. The encapsulating user generates a secret key and uses the decapsulating user's identity to encrypt the secret key to the decapsulating user, and the decapsulating user obtains the secret key through the decapsulation process with the private key.

This part also describes an identity-based public key encryption algorithm built upon pairings on elliptic curves. The public key encryption algorithm is constructed from the combination of the key encapsulation mechanism described above together with a data encapsulation mechanism to provide data confidentiality. There are two types of data encapsulation mechanisms: stream ciphers based on a key derivation function, and block ciphers combined with a key derivation function. For the identity-based encryption algorithm, the decrypting entity holds an identity and the corresponding private key, which is generated by the KGC using the master private key and the identity of decrypting entity. The encrypting entity encrypts data with the decrypting entity's identity, and the decrypting entity decrypts the data with its private key.

5.2 System parameters

The system parameters include: the curve identifier cid, the parameters of the elliptic curve base field F_q, parameters a and b of the elliptic curve equation, the twisted curve parameter β (if the least significant 4 bits of cid is 2), the prime factor N of the order of the curve and the cofactor cf relative to N, the embedding degree k of the curve $E(F_q)$ relative to N, a generator P_1 of the cyclic subgroup $\mathbb{G}_1$ of $E(F_{q^{d_1}})$ of order N (where d_1 divides k), a generator P_2 of the cyclic subgroup of $E(F_{q^{d_2}})$ of order N (where d_2 divides k), the identifier eid of bilinear pairing e, and (optional) the homomorphism Ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$.

The range of the bilinear pairing e is the multiplicative cyclic group $\mathbb{G}_T$ of order N.

For detailed descriptions of the system parameters as well as their verification, please refer to Clause 7 of GM/T 0044.1–2016.

5.3 Generation of the encryption master key and the user's encryption private key

The KGC generates a random number $ke \in [1, N-1]$ as the encryption master private key, computes the element $P_{pub-e} = [ke]P_1$ in $\mathbb{G}_1$ as the encryption master public key, and then the encryption master key pair is (ke, P_{pub-e}). The KGC keeps ke secret and makes P_{pub-e} public.

The KGC selects a one-byte encryption private key generating function identified by the identifier hid, and makes it public.

Let ID_B denote the identity of user B. To generate the encryption private key de_B of B, the KGC first computes $t_1 = H_1(ID_B||hid, N) + ke$ over the finite field F_N. If $t_1 = 0$, it regenerates the encryption master private key, computes the encryption master public key and makes it public, and updates the encryption private keys of existing users. Otherwise, it computes $t_2 = ke \cdot t_1^{-1}$, and then computes $d_{eB} = [t_2]P_2$.

5.4 Auxiliary functions

5.4.1 Overview

Five types of auxiliary functions are used in the identity-based key encapsulation mechanism and the public key encryption algorithm specified in this part: cryptographic hash functions, key derivation functions, message authentication code functions, random number generators and block cipher algorithms. The security of the key encapsulation mechanism and the public key encryption algorithm is directly impacted by these auxiliary functions.

5.4.2 Cryptographic hash functions

5.4.2.1 Cryptographic hash function $H_v()$

The output of the cryptographic hash function $H_v()$ is a v-bit hash value. This part adopts the cryptographic hash function specified in GM/T 0004 SM3 cryptographic hash algorithm.

5.4.2.2 Cryptographic function $H_1()$

The input of the cryptographic function $H_1(Z, n)$ is a bit string Z and an integer n, and its output is an integer $h_1 \in [1, n-1]$. $H_1(Z, n)$ invokes the cryptographic hash function $H_v()$ internally. $H_v()$ is specified in 5.4.2.1.

Cryptographic function $H_1(Z, n)$:

Input: a bit string Z and an integer n.

Output: an integer $h_1 \in [1, n-1]$.

```
Step 1: Initialize a 32-bit counter $ct = 0x00000001$;

Step 2: Compute $hlen = 8 \times \lceil (5 \times (\log_2 n))/32 \rceil$;

Step 3: For $i = 1$ to $\lceil hlen/v \rceil$:

    Step 3.1: Compute $Ha_i = H_v(0x01||Z||ct)$;

    Step 3.2: $ct$++;

Step 4: If $hlen/v$ is an integer, set $Ha!_{\lceil hlen/v \rceil} = Ha_{\lceil hlen/v \rceil}$. Otherwise, set $Ha!_{\lceil hlen/v \rceil}$ to be the leftmost $(hlen - (v \times \lfloor hlen/v \rfloor))$ bits of $Ha_{\lceil hlen/v \rceil}$.

Step 5: Set $Ha = Ha_1||Ha_2||\cdots|| Ha_{\lceil hlen/v \rceil - 1} ||Ha!_{\lceil hlen/v \rceil}$. Convert the data type of $Ha$ to integer as specified in Clauses 6.2.4 and 6.2.3 of GM/T 0044.1–2016.

Step 6: Compute $h_1 = (Ha \bmod (n-1)) + 1$.
```

5.4.2.3 Cryptographic function $H_2()$

The input of the cryptographic function $H_2(Z, n)$ is a bit string Z and an integer n, and its output is an integer $h_2 \in [1, n-1]$. $H_2(Z, n)$ invokes the cryptographic hash function $H_v()$ internally. $H_v()$ is specified in 5.4.2.1.

Cryptographic function $H_2(Z, n)$:

Input: a bit string Z, an integer n.

Output: an integer $h_2 \in [1, n-1]$.

```
Step 1: Initialize a 32-bit counter $ct = 0x00000001$;

Step 2: Compute $hlen = 8 \times \lceil (5 \times (\log_2 n))/32 \rceil$;

Step 3: For $i = 1$ to $\lceil hlen/v \rceil$:

    Step 3.1: Compute $Ha_i = H_v(0x02||Z||ct)$;

    Step 3.2: $ct$++;

Step 4: If $hlen/v$ is an integer, set $Ha!_{\lceil hlen/v \rceil} = Ha_{\lceil hlen/v \rceil}$. Otherwise, set $Ha!_{\lceil hlen/v \rceil}$ to be the leftmost $(hlen - (v \times \lfloor hlen/v \rfloor))$ bits of $Ha_{\lceil hlen/v \rceil}$.

Step 5: Set $Ha = Ha_1||Ha_2||\cdots|| Ha_{\lceil hlen/v \rceil - 1} ||Ha!_{\lceil hlen/v \rceil}$. Convert the data type of $Ha$ to integer as specified in Clauses 6.2.4 and 6.2.3 of GM/T 0044.1–2016.

Step 6: Compute $h_2 = (Ha \bmod (n-1)) + 1$.
```

5.4.3 Key derivation functions

The key derivation functions adopted in this part are specified in Clause 5.4.3 of GM/T 0044.3–2016.

5.4.4 Block cipher algorithms

A block cipher algorithm is comprised of an encryption algorithm $Enc(K_1, m)$ and a decryption algorithm $Dec(K_1, c)$. $Enc(K_1, m)$ uses the secret key K_1 to encrypt the plaintext m and outputs the ciphertext c. $Dec(K_1, c)$ uses the secret key K_1 to decrypt the ciphertext c and outputs the plaintext m or reports an error. The bit length of K_1 is denoted by K_1_len.

This part adopts the block cipher algorithm specified in GM/T 0002 SM4 block cipher algorithm.

5.4.5 Message authentication code functions

The aim of the message authentication code function $MAC(K_2, Z)$ is to protect the message Z from unauthorized modifications. The message authentication code of message Z is generated under the control of K_2. The bit length of K_2 is denoted by K_2_len. For the identity-based encryption algorithm in this part, the message authentication code function uses the key generated by the key derivation function to obtain the message authentication code of the ciphertext, allowing the decrypting entity to identify the message source and verify integrity of the message.

The message authentication code functions invoke the cryptographic hash functions.

Let $H_v(\,)$ be a cryptographic hash function and its output is a hash value of length v bits long.

Message authentication code function $\boldsymbol{MAC(K_2, Z)}$:

Input: a bit string K_2 (a key of length K_2_len bits) and a bit string Z (the message to be processed to obtain MAC).

Output: a bit string K of length v (the MAC of the message Z).

Step 1: $K = H_v(Z||K_2)$.

5.4.6 Random number generators

This part adopts the random number generators conforming to GM/T 0005.

6 Key encapsulation mechanism and its process

6.1 Key encapsulation algorithm and its process

6.1.1 Key encapsulation algorithm

To encapsulate a key of length $klen$ to user B, the encapsulating entity user A shall perform the following steps.

A1: Compute $Q_B = [H_1(ID_B||hid, N)]P_1 + P_{pub-e} \in \mathbb{G}_1$.

A2: Generate a random integer $r \in [1, N-1]$.

A3: Compute $C = [r]Q_B$ of $\mathbb{G}_1$, and convert the data type of C to bit string as specified in Clauses 6.2.8 and 6.2.5 of GM/T 0044.1–2016.

A4: Compute $g = e(P_{pub-e}, P_2)$ of $\mathbb{G}_T$.

A5: Compute $w = g^r$ of $\mathbb{G}_T$, and convert the data type of w to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1–2016.

A6: Compute $K = KDF(C||w||ID_B, klen)$, if $K = 0$, go to A2.

A7: Output (K, C), where K is the encapsulated key, C is the encapsulated ciphertext.

6.1.2 Key encapsulation process

The key encapsulation process is shown in Figure 1.

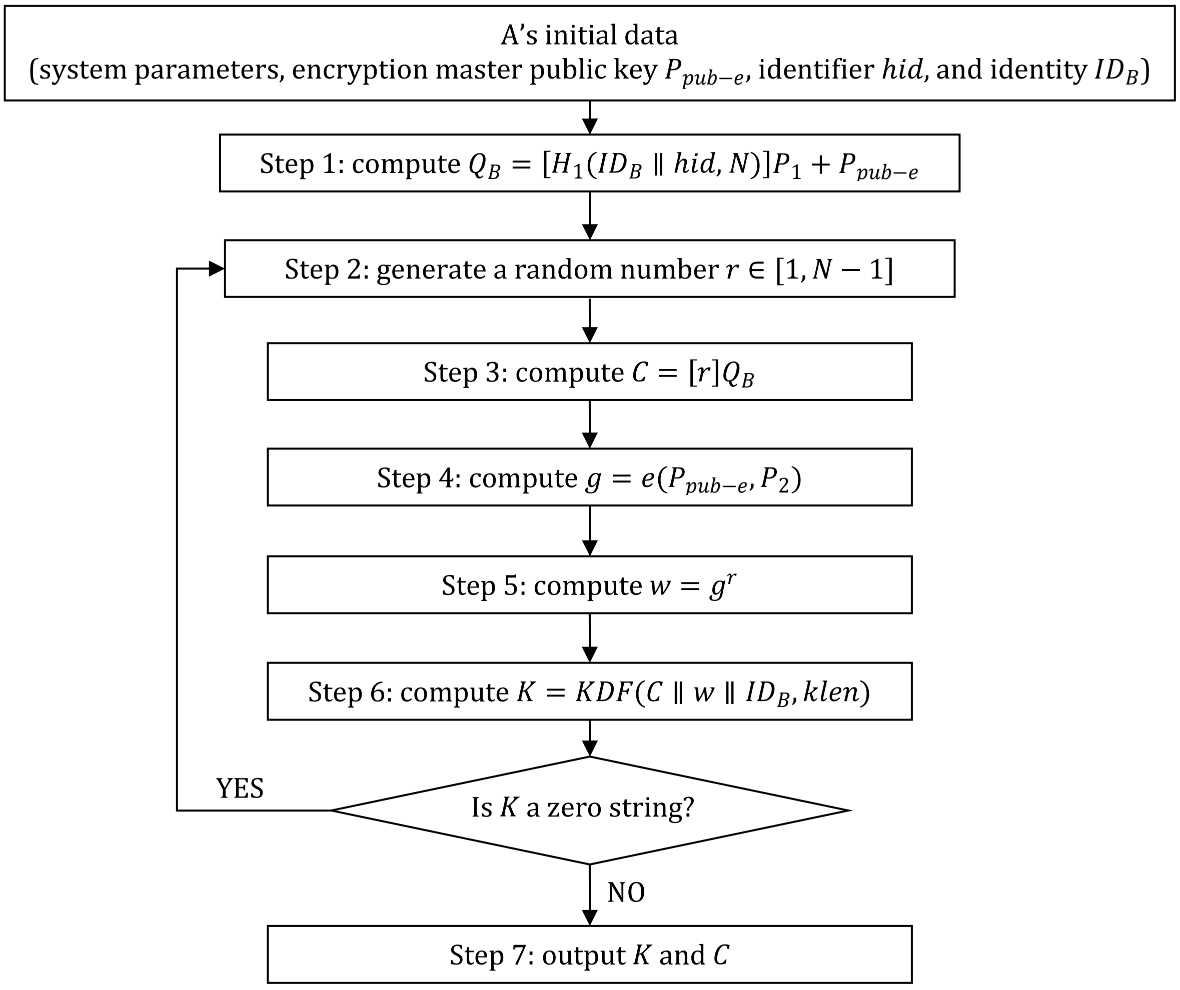

A's initial data
(system parameters, encryption master public key P_{pub-e}, identifier hid, and identity ID_B)
Step 1: compute $Q_B = [H_1(ID_B \parallel hid, N)]P_1 + P_{pub-e}$
Step 2: generate a random number $r \in [1, N-1]$
Step 3: compute $C = [r]Q_B$
Step 4: compute $g = e(P_{pub-e}, P_2)$
Step 5: compute $w = g^r$
Step 6: compute $K = KDF(C \parallel w \parallel ID_B, klen)$
YES
Is K a zero string?
NO
Step 7: output K and C

Figure 1: Key encapsulation process

6.2 Key decapsulation algorithm and its process

6.2.1 Decapsulation algorithm

After user B receives the ciphertext C, in order to decapsulate K, B shall perform the following steps.

B1: Verify that $C \in G_1$ as specified in Clause 4.5 of GM/T 0044.1–2016. If not, report an error and exit.

B2: Compute $w' = e(C, de_B)$ of $\mathbb{G}_T$, and convert the data type of w' to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1−2016.

B3: Convert the data type of C to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1−2016, and compute $K' = KDF(C||w'||ID_B, klen)$. If $K' = 0$, report an error and exit.

B4: Output K'.

6.2.2 Key decapsulation process

The key decapsulation process is shown in Figure 2.

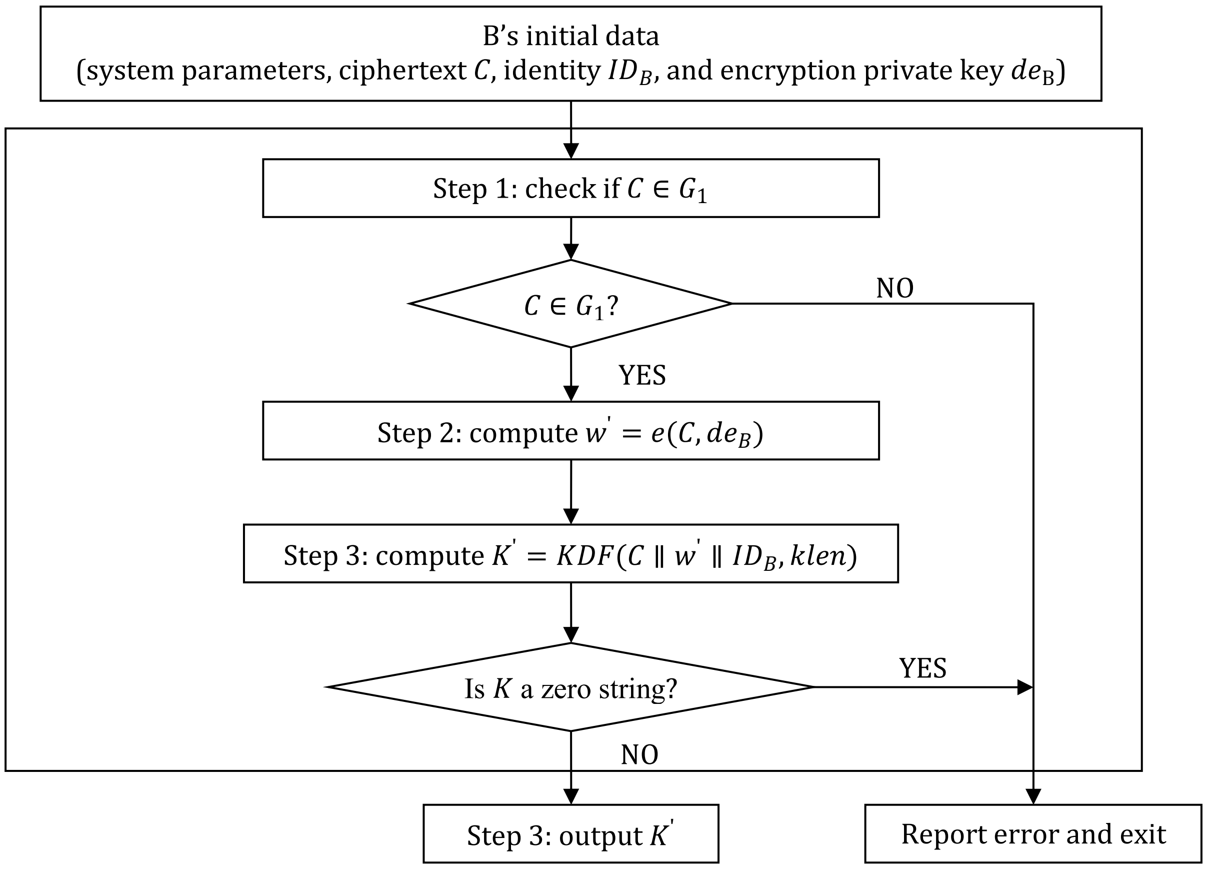

B's initial data
(system parameters, ciphertext C, identity ID_B, and encryption private key de_B)
Step 1: check if $C \in G_1$
$C \in G_1$?
NO
YES
Step 2: compute $w' = e(C, de_B)$
Step 3: compute $K' = KDF(C \parallel w' \parallel ID_B, klen)$
Is K a zero string?
YES
NO
Step 3: output K'
Report error and exit

Figure 2: Key decapsulation process

7 Public key encryption algorithm and its process

7.1 Encryption algorithm and its process

7.1.1 Encryption algorithm

Let M be the message to be sent, $mlen$ the bit length of M. K_1_len is the bit length of the key K_1 used with the block cipher. K_2_len is the bit length of the key K_2 for $MAC(K_2, Z)$.

In order to encrypt a message M to user B, user A shall perform the following steps.

A1: Compute $Q_B = [H_1(ID_B||hid, N)]P_1 + P_{pub-e} \in \mathbb{G}_1$.

A2: Generate a random integer $r \in [1, N-1]$.

A3: Compute $C_1 = [r]Q_B$ of $\mathbb{G}_1$, and convert the data type of C_1 to bit string as specified in Clauses 6.2.8 and 6.2.5 of GM/T 0044.1−2016.

A4: Compute $g = e(P_{pub-e}, P_2)$ of $\mathbb{G}_T$.

A5: Compute $w = g^r$ of $\mathbb{G}_T$, and convert the data type of w to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1−2016.

A6: Compute according to the type of encryption algorithm:

a) Stream cipher based on the key derivation function
   1) Compute $klen = mlen + K_2\_len$, $K = KDF(C_1||w||ID_B, klen)$. Let K_1 be the leftmost $mlen$ bits of K, and K_2 be the remaining K_2_len bits. If $K_1 = 0$, go to A2.
   2) Compute $C_2 = M \oplus K_1$.
b) Block cipher combined with the key derivation function
   1) Compute $klen = K_1\_len + K_2\_len$, $K = KDF(C_1||w||ID_B, klen)$. Let K_1 be the leftmost K_1_len bits of K, and K_2 be the remaining K_2_len bits. If $K_1 = 0$, go to A2.
   2) Compute $C_2 = Enc(K_1, M)$. If the working mode of Enc requires message padding, the l bytes message M shall be padded in the following manner: set $k = K_1\_len/8$, M shall be padded with $k - l \bmod k$ bytes of value $k - l \bmod k$. If Enc requires an initial vector, it shall set the value all zeros.

A7: Compute $C_3 = MAC(K_2, C_2)$.

A8: Output ciphertext $C = C_1||C_3||C_2$.

7.1.2 Encryption process

The encryption process is shown in Figure 3.

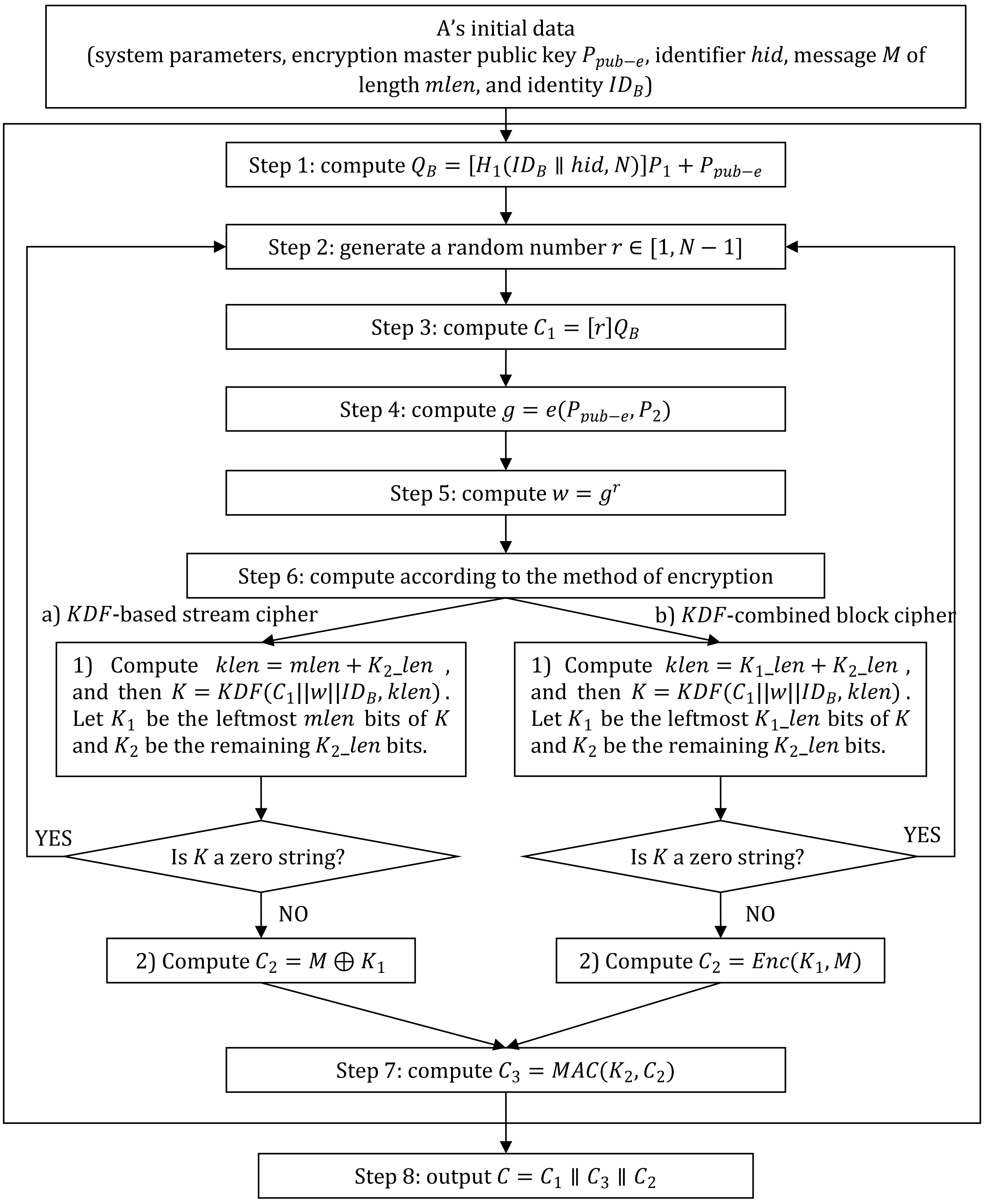

A's initial data (system parameters, encryption master public key P_{pub-e}, identifier hid, message M of length $mlen$, and identity ID_B)
Step 1: compute $Q_B = [H_1(ID_B \parallel hid, N)]P_1 + P_{pub-e}$
Step 2: generate a random number $r \in [1, N-1]$
Step 3: compute $C_1 = [r]Q_B$
Step 4: compute $g = e(P_{pub-e}, P_2)$
Step 5: compute $w = g^r$
Step 6: compute according to the method of encryption
a) KDF-based stream cipher
b) KDF-combined block cipher
1) Compute $klen = mlen + K_2\_len$, and then $K = KDF(C_1||w||ID_B, klen)$. Let K_1 be the leftmost $mlen$ bits of K and K_2 be the remaining K_2_len bits.
1) Compute $klen = K_1\_len + K_2\_len$, and then $K = KDF(C_1||w||ID_B, klen)$. Let K_1 be the leftmost K_1_len bits of K and K_2 be the remaining K_2_len bits.
YES
Is K a zero string?
Is K a zero string?
YES
NO
NO
2) Compute $C_2 = M \oplus K_1$
2) Compute $C_2 = Enc(K_1, M)$
Step 7: compute $C_3 = MAC(K_2, C_2)$
Step 8: output $C = C_1 \parallel C_3 \parallel C_2$

Figure 3: Encryption process

7.2 Decryption algorithm and its process

7.2.1 Decryption algorithm

Let $mlen$ be the bit length of C_2 of ciphertext $C = C_1||C_3||C_2$. K_1_len is the bit length of the key K_1 used with the block cipher. K_2_len is the bit length of the key K_2 for $MAC(K_2, Z)$.

To decrypt C, user B needs to perform the following steps.

B1: Extract bit string C_1 from C. Convert the data type of C_1 to a point on elliptic curve as specified in Clauses 6.2.4 and 6.2.9 of GM/T 0044.1–2016. Verify $C_1 \in \mathbb{G}_1$ as specified in Clause 4.5 of GM/T 0044.1–2016; if not, report an error and exit.

B2: Compute $w' = e(C_1, de_B)$ of $\mathbb{G}_T$, and convert the data type of w' to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1–2016.

B3: Compute according to the type of encryption algorithm:

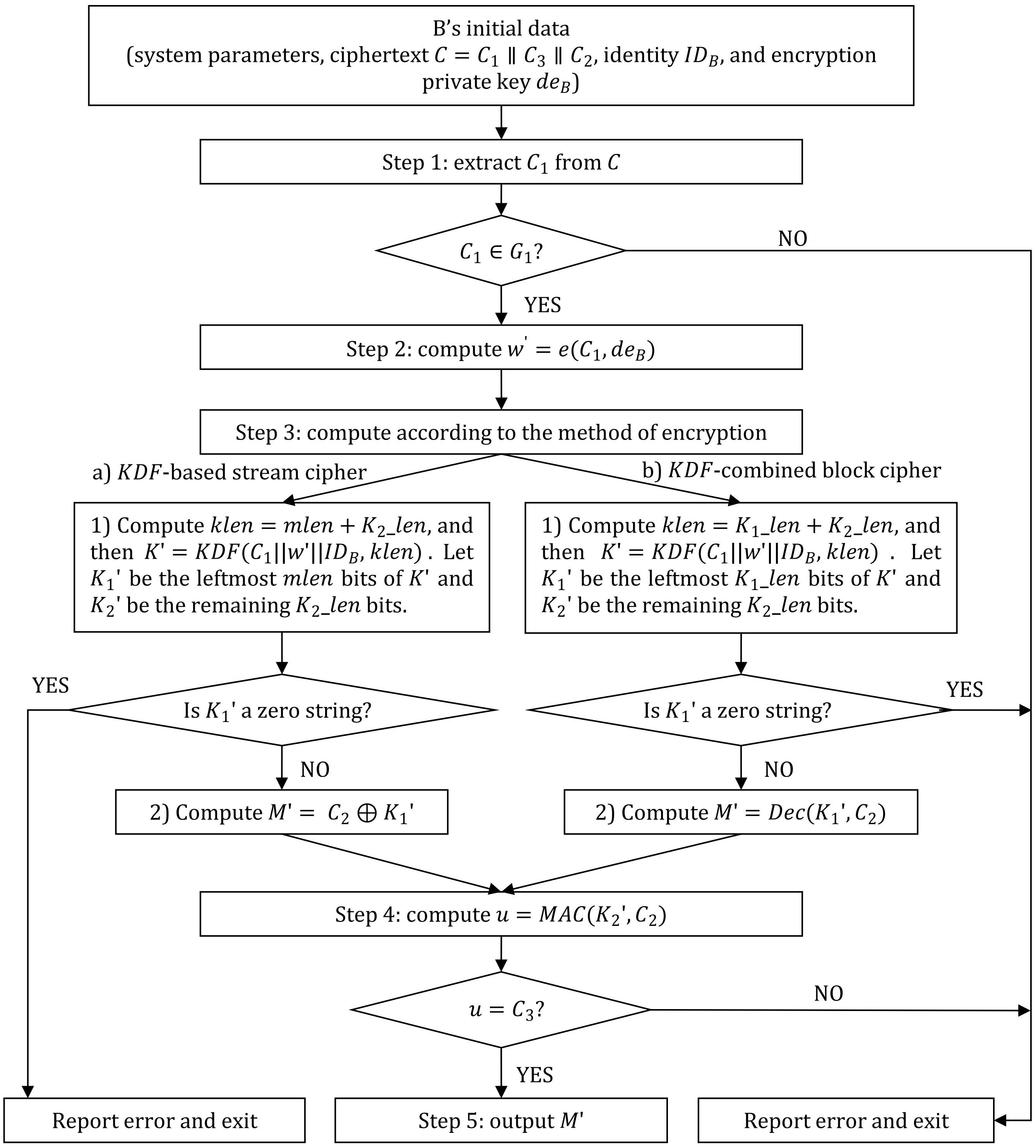
a) Stream cipher based on the key derivation function
   1) Compute $klen = mlen + K_2\_len$, $K' = KDF(C_1||w'||ID_B, klen)$. Let K_1' be the leftmost $mlen$ bits of K', and K_2' be the remaining K_2_len bits. If $K_1' = 0$, report an error and exit;
   2) Compute $M' = C_2 \oplus K_1'$.
b) Block cipher combined with the key derivation function
   1) Compute $klen = K_1\_len + K_2\_len$, $K' = KDF(C_1||w'||ID_B, klen)$. Let K_1' be the leftmost K_1_len bits of K', and K_2' be the remaining K_2_len bits. If $K_1' = 0$, report an error and exit;
   2) Compute $M' = Dec(K_1', C_2)$.

B4: Compute $u = MAC(K_2', C_2)$. Extract bit string C_3 from C; if $u \neq C_3$, report an error and exit;

B5: Output plaintext M'.

7.2.2 Decryption process

The decryption process is shown in Figure 4.



B's initial data (system parameters, ciphertext $C = C_1 \parallel C_3 \parallel C_2$, identity ID_B, and encryption private key de_B)
Step 1: extract C_1 from C
$C_1 \in G_1$?
NO
YES
Step 2: compute $w' = e(C_1, de_B)$
Step 3: compute according to the method of encryption
a) KDF-based stream cipher
b) KDF-combined block cipher
1) Compute $klen = mlen + K_2\_len$, and then $K' = KDF(C_1||w'||ID_B, klen)$. Let K_1' be the leftmost $mlen$ bits of K' and K_2' be the remaining K_2_len bits.
1) Compute $klen = K_1\_len + K_2\_len$, and then $K' = KDF(C_1||w'||ID_B, klen)$. Let K_1' be the leftmost K_1_len bits of K' and K_2' be the remaining K_2_len bits.
YES
Is K_1' a zero string?
Is K_1' a zero string?
YES
NO
NO
2) Compute $M' = C_2 \oplus K_1'$
2) Compute $M' = Dec(K_1', C_2)$
Step 4: compute $u = MAC(K_2', C_2)$
$u = C_3$?
NO
YES
Report error and exit
Step 5: output M'
Report error and exit

Figure 4: Decryption process