SM9 identity-based cryptographic algorithms

Part 3: Key exchange protocol

Contents

SM9 identity-based cryptographic algorithms

Part 3: Key exchange protocol

1 Scope

This part of GM/T 0044−2016 describes an identity-based key exchange protocol built upon pairings on elliptic curves, and specifies the corresponding processes. This protocol enables two communication entities to compute a shared secret key, which is generated from input provided by both entities, as the result of two or (optionally) three message exchanges over the identity of the peer entity and its own private key. The shared secret key can be used as the session key of symmetric cryptographic algorithms. The optional message exchange step in the protocol allows for key confirmation.

This part applies to key management and agreement.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes 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 0004–2012, SM3 cryptographic hash algorithm

GM/T 0044.1–2016, SM9 identity-based cryptographic algorithms — Part 1: General

3 Terms and definitions

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

3.1 key exchange

scheme to exchange keys securely between communication entities, enables both entities to exchange keys securely for transmitting information over an insecure communication channel

3.2 key agreement

process to generate a shared secret key among multiple users, where no user can determinate the value of the key beforehand

3.3 key confirmation from A to B

assurance for user B that user A is in possession of the correct key

3.4 key derivation function

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

3.5 initiator

entity which initiates the first round of message exchange in the protocol

3.6 responder

entity that does not initiate the first round of message exchange in the protocol

3.7 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 generators, while the encryption master public key is generated with the encryption master private key and the system parameters

3.8 identity

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

3.9 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: 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 (a 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

de_A: the encryption private key of the user A

de_B: the encryption private key of the user B

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 $g^u = \underbrace{g \cdot g \cdot \cdots \cdot g}_{u\ g's}$

$H_v()$: cryptographic hash functions

$H_1(), H_2()$: 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_A: the identity of the user A that uniquely determines the public key of A

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

$KDF()$: the key derivation 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 $\mathbb{G}_2$

r_A: the temporary key generated by the user A during the key exchange

r_B: the temporary key generated by the user B during the key exchange

SK_A, SK_B: the shared secret key agreed in the key exchange protocol

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 $\mathbb{G}_2$

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

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

$x \parallel y$: 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

β: the twisted curve parameter

5 Algorithm parameters and auxiliary functions

5.1 Overview

This part describes an identity-based key exchange protocol implemented upon pairings of elliptic curves. Both the initiator, user A, and the responder, user B, participates in the key exchange protocol with its identity and its corresponding encryption private key, which is generated by the KGC with the

encryption master private key and the user's identity. A and B can negotiate a secret key only known to themselves with their identities and encryption private keys through exchanging messages. Both users can perform key confirmation of its counterpart through an optional message exchange. The shared secret key is often used in some symmetric cryptographic algorithm. This key exchange protocol can be used for key management and key agreement.

5.2 System parameters

The system parameters include: the curve identifier cid, the parameters of the elliptic curve base field F_q, the parameters of the elliptic curve equation a and b, 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 bilinear pairing identifier eid of e, and optionally 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 chooses a one-byte encryption private key generating function identifier hid and makes it public.

Let ID_A and ID_B denote the identities of the users A and B respectively. To generate the encryption private key de_A of A, the KGC first computes $t_1 = H_1(ID_A \parallel hid, N) + ke$ 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 existing encryption private keys of users. Otherwise, it computes $t_2 = ke \cdot t_1^{-1}$, and then computes $de_A = [t_2]P_2$. To generate the encryption private key de_B of B, the KGC first computes $t_3 = H_1(ID_B \parallel hid, N) + ke$ over the finite field F_N. If $t_3 = 0$, it regenerates the encryption master private key, computes the encryption master public key and makes it public, and updates the existing encryption private keys of users. Otherwise, it computes $t_4 = ke \cdot t_3^{-1}$, and then computes $de_B = [t_4]P_2$.

5.4 Auxiliary functions

5.4.1 Overview

There are three types of auxiliary functions used in the identity-based key exchange protocol specified in this part: cryptographic hash functions, key derivation functions and random number generators. The security of the key exchange protocol 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 functions approved by the State Cryptography Administration such as the 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 function is used to derive keys from a shared secret bit string. In the key agreement process, the key derivation function takes the shared secret bit string obtained in the key exchange process as input, and generates session keys or other secret keys for further encryption.

The key derivation functions invokes cryptographic hash functions specified above.

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

Key derivation function $KDF(Z, klen)$**:**

Input: a bit string Z (shared by both entities) and an integer $klen$ (denotes the required bit length of secret keys, and $klen < (2^{32} - 1)v$).

Output: a bit string K of length $klen$.

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

Step 2: For $i = 1$ to $\lceil klen/v \rceil$:

Step 2.1: Compute $Ha_i = H_v(Z \parallel ct)$;

Step 2.2: $ct++$;

Step 3: If $klen/v$ is an integer, then set $Ha!_{\lceil klen/v \rceil} = Ha_{\lceil klen/v \rceil}$. Otherwise, set $Ha!_{\lceil klen/v \rceil}$ be the leftmost $\left(klen - (v \times \lfloor klen/v \rfloor)\right)$ bits of $Ha_{\lceil klen/v \rceil}$;

Step 4: Set $K = Ha_1 \parallel Ha_2 \parallel \cdots \parallel Ha_{\lceil klen/v \rceil - 1} \parallel Ha!_{\lceil klen/v \rceil}$.

5.4.4 Random number generators

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

6 Key exchange protocol and its process

6.1 Key exchange protocol

Assume that the users A and B are negotiating a bit string of length $klen$, where A is the initiator and B is the responder.

In order to obtain the same keys, both A and B shall perform the following steps.

User A:

A1: Compute the element $Q_B = [H_1(ID_B \parallel hid, N)]P_1 + P_{pub-e}$ over $\mathbb{G}_1$;

A2: Generate a random number $r_A \in [1, N-1]$;

A3: Compute the element $R_A = [r_A]Q_B$ over $\mathbb{G}_1$;

A4: Send R_A to B;

User B:

B1: Compute the element $Q_A = [H_1(ID_A \parallel hid, N)]P_1 + P_{pub-e}$ over $\mathbb{G}_1$;

B2: Generate a random number $r_B \in [1, N-1]$;

B3: Compute the element $R_B = [r_B]Q_A$ over $\mathbb{G}_1$;

B4: Verify $R_A \in \mathbb{G}_1$ as specified in Clause 4.5 of GM/T 0044.1–2016. If not, the protocol fails. Otherwise, compute over $\mathbb{G}_T$: $g_1 = e(R_A, de_B)$, $g_2 = e\left(P_{pub-e}, P_2\right)^{r_B}$, $g_3 = g_1^{r_B}$. Convert the data type of g_1, g_2, g_3 to bit string as specified in Clauses 6.2.8 and 6.2.5 of GM/T 0044.1–2016.

B5: Convert the data type of R_A and R_B to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1–2016, and compute $SK_B = KDF(ID_A \parallel ID_B \parallel R_A \parallel R_B \parallel g_1 \parallel g_2 \parallel g_3, klen)$.

B6: (Optional) Compute $S_B = H_v\left(0x82 \parallel g_1 \parallel H_v(g_2 \parallel g_3 \parallel ID_A \parallel ID_B \parallel R_A \parallel R_B)\right)$.

B7: Send R_B and (optionally) S_B to A.

User A:

A5: Verify $R_B \in \mathbb{G}_1$ as specified in Clause 4.5 of GM/T 0044.1–2016. If not, the protocol fails. Otherwise, compute over $\mathbb{G}_T$: $g_1' = e\left(P_{pub-e}, P_2\right)^{r_A}$, $g_2' = e(R_B, de_A)$, $g_3' = (g_2')^{r_A}$. Convert the data type of g_1', g_2', g_3' to bit string as specified in Clauses 6.2.6 and 6.2.5 of GM/T 0044.1–2016.

A6: Convert the data type of R_A and R_B to bit string as specified in Clauses 6.2.8 and 6.2.5 of GM/T 0044.1–2016, and (optionally) compute $S_1 = H_v\left(0x82 \parallel g_1' \parallel H_v(g_2' \parallel g_3' \parallel ID_A \parallel ID_B \parallel R_A \parallel R_B)\right)$, and verify if $S_1 = S_B$, if not, the key confirmation from B to A fails.

A7: Compute $SK_A = KDF(ID_A \parallel ID_B \parallel R_A \parallel R_B \parallel g_1' \parallel g_2' \parallel g_3', klen)$.

A8: (Optional) Compute $S_A = H_v\left(0x83 \parallel g_1' \parallel H_v(g_2' \parallel g_3' \parallel ID_A \parallel ID_B \parallel R_A \parallel R_B)\right)$ and send S_A to B.

User B:

B8: (Optional) Compute $S_2 = H_v\left(0x83 \parallel g_1 \parallel H_v(g_2 \parallel g_3 \parallel ID_A \parallel ID_B \parallel R_A \parallel R_B)\right)$, and verify if $S_2 = S_A$, if not, the key confirmation from A to B fails.

6.2 Key exchange process

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

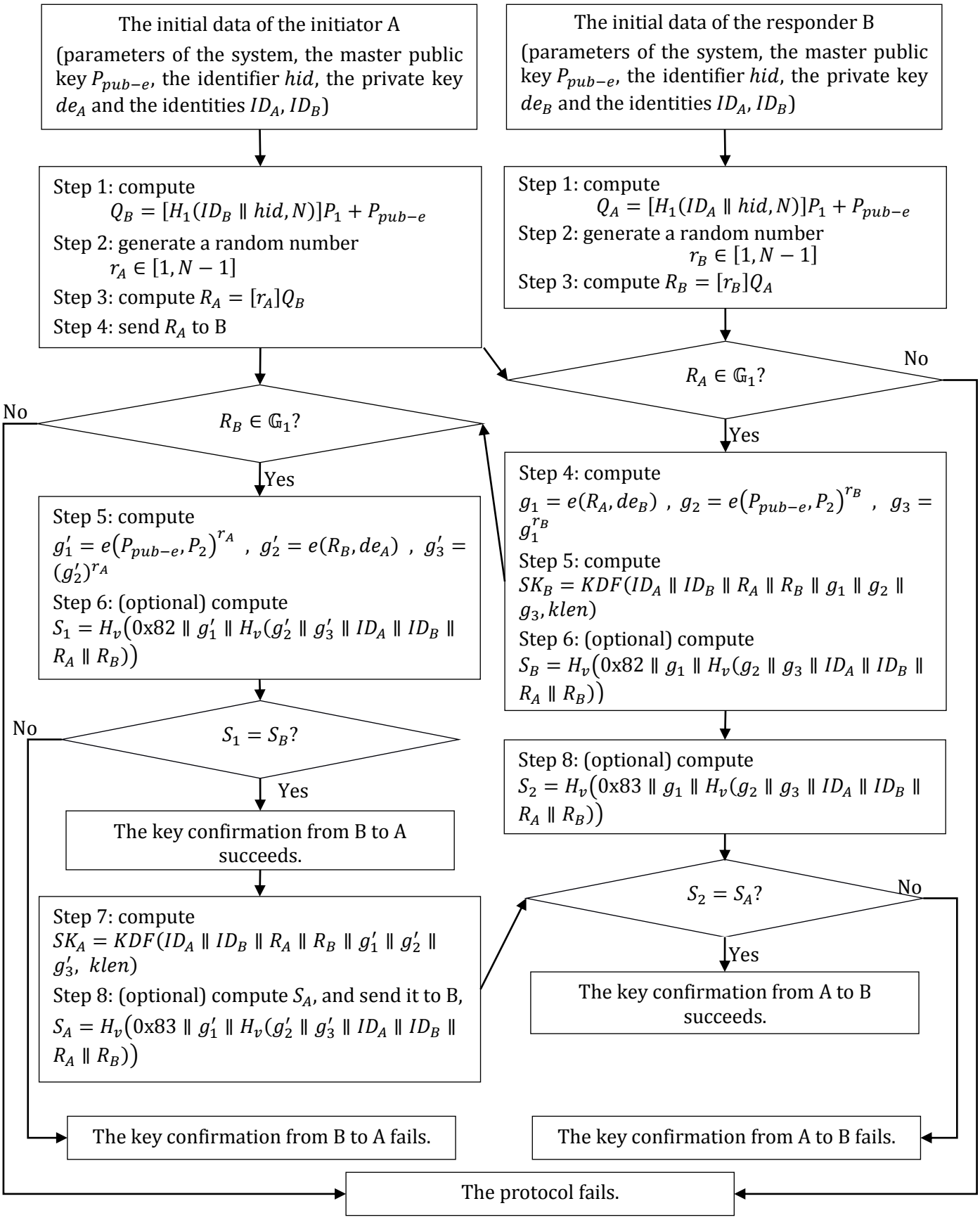

Figure 1: Key exchange protocol process