The GmSSL Project

支持国密SM2/SM3/SM4/SM9的密码工具箱

Fork me on GitHub

SM9 Digital Signature

System Paremeters

The system parameter set consists of the curve identifier cid; the parameters of the elliptic curve base filed F_q ; the parameters a and b of the elliptic curve equation; the parameterβ of the curve (if the lower 4 bits of the cid are 2);The prime factor of the curve order N and the remainder factor cf with respect to N;The number of embedding times k of the curve $E(F_q)$ with respect to N; the generator P_1 of the N-th order cyclic subgroup G_1 of $E(F_{q^{d_1}})$ (divides k by d_1);agenerator P_2 of an N-th order cyclic subgroup G_2 of $E(F_{q^{d_2}})$ (divides k by d_2); bilinear pair identifier of e eid; (Optional)Homomorphism of G_2 to G_1 ψ.

The range of the bilinear pair e is an order-N multiplicative cyclic group G_T.

System Signature Master Key and User Signature Key Generation

KGC generates random number $ks \in [1, N-1]$as the signature master-private key and calculates the element $P_{pub-s} = [ks]P_2$ in G_2 as the signature master public key. The signature master key pair is (ks, P_{pub-s})。KGC secretly save ks, and set P_{pub-s} public.

KGC chooses and exposes the signed private key generate function identifier hid represented by one byte.

The ID of user A is ID_A. To generate the private key ds_A of user A, KGC first calculates $t_1 = H_1(ID_A \| hid, N) + ks$ on the finite field F_N. If $t_1 = 0$ we need to re-sign the signature of the main private key,calculate and public signature master public key, and update the signature private key of the existing user; otherwise calculate

$t_2 = ks \cdot t_1^{-1} mod N$and then calculate $ds_A = [t_2]P_1$.

SM9 Digital Signature Generation Algorithm

The message to be signed is a bit string M, and in order to obtain the digital signature (h, S) of the message M, the user A who is the signer should implement the following operation steps:

1. Calculate the element $g = e(P_1, P_{pub-s})$ in the group $\mathbb{G}_T$;
2. Generate a random number $r \in [1, N-1]$;
3. Calculate element $w = g^r$ in group G_T, convert data type of w into bit string;
4. Calculate integer$h = H_2(M \| w, N)$;
5. Calculate an integer$l = (r - h) mod N$, return 2 if l = 0;
6. Compute elements $S = [l]ds_A$ in group G_1 ;
7. The signature of message M is (h, S)。

SM9 Digital Signature Verification Algorithm

In order to check the received message $M^‘$ ‘and its digital signature $(h^‘, S^‘)$, the user B as the verifier should implement the following operation steps:

1. Check whether $h^‘ \in [1, N-1]$ holds; Then the verification fails;
2. Convert the data type of $S^‘$ to a point on the elliptic curve to test whether $S^‘ \in G_1$ holds; if not, the verification fails;
3. Calculate the element $g = e(P_1, P_{pub-s})$ in the groupG_T ;
4. Compute elements $t = g^{h^‘}$ in group G_T;
5. Calculate integer $h_1 = H_1(ID_A \| hid, N)$;
6. Calculate elements $P = [h_1]P_2 + P_{pub-s}$ in group G_2;
7. Calculate the element $u = e(S^‘, P)$ in group G_T;
8. Calculate the element $w^‘ = u\dot{t}$ in group G_T, convert the data type of $w^‘$ to a bit string;
9. Calculate the integer $h_2 = H_2(H^‘ \| w^‘, N)$, and test if $h_2 = h^‘$. If yes, then the verification is successful; otherwise, the verification fails

SM9 Key Exchange

System Paremeters

The system parameter set consists of the curve identifier cid; the parameters of the elliptic curve base filed F_q ; the parameters a and b of the elliptic curve equation; the parameterβ of the curve (if the lower 4 bits of the cid are 2);The prime factor of the curve order N and the remainder factor cf with respect to N;The number of embedding times k of the curve $E(F_q)$ with respect to N; the generator P_1 of the N-th order cyclic subgroup G_1 of $E(F_{q^{d_1}})$ (divides k by d_1);agenerator P_2 of an N-th order cyclic subgroup G_2 of $E(F_{q^{d_2}})$ (divides k by d_2); bilinear pair identifier of e eid; (Optional)Homomorphism of G_2 to G_1 ψ.

The range of the bilinear pair e is an order-N multiplicative cyclic group G_T.

System Encryption Master Key and User Encryption Key Generation

KGC generates random number $ks \in [1, N-1]$as the signature master-private key and calculates the element $P_{pub-s} = [ks]P_2$ in G_2 as the signature master public key. The signature master key pair is (ks, P_{pub-s})。KGC secretly save ks, and set P_{pub-s} public.

KGC chooses and exposes the signed private key generate function identifier hid represented by one byte.

The identities of users A and B are ID_A and ID_B respectively. To generate the encrypted private key de_A of user A, KGC first calculates $t_1 = H_1(ID_A||hid, N) + ke$ on the finite field F_N, and if t1 = 0, it needs to regenerate the encrypted master private key to calculate and disclose the master public encrypt key, and update the private encrypt key of the existing user; otherwise calculate $t_2 = ks \cdot t_1^{-1} modN$ and then calculate $de_A = [t_2]P_2$. In order to generate the encrypted private key de_B of user B, KGC first computes$t_3 = H_1(ID_B||hid, N) + ke$ on the finite field F_N, if $t_3 = 0$, it needs to regenerate the encrypted master private key to calculate and disclose the master public encrypt key, and update the private encrypt key of the existing user; otherwise, $t_4 = ke \cdot t_3^{-1} modN$is calculated and then $de_B = [t_4]P_2$ is calculated.

Key Exchange Protocol And Process

Suppose that the length of the key data obtained through negotiation between users A and B is klen bits, user A is the initiator, and user B is the responder. In order to obtain the same key, both users A and B should implement the following operation steps: User A:

1. Calculate element $Q_B = [H_1(ID_B \| hid, N)]P_1 + P_{pub-e}$ in group G_1;
2. Generate a random number$r_a \in [1, N-1]$;
3. Calculate element $R_A = [r_A]Q_B$ in group G_1;
4. Send R_A to user B;

User B:

1. Calculate element $Q_A = [H_1(ID_A \| hid, N)]P_1 + P_{pub-e}$ in group G_1;
2. Generate a random number$r_B \in [1, N-1]$;
3. Calculate the element $R_B = [r_B]Q_A$ in group G_1 ;
4. Verify whether $R_A \in G_1$ is true, and if not, the negotiation fails; otherwise, calculate the element $g_1 = e(R_A, de_B), g_2 = e(P_{pub-e}, P_2)^{rB}, g_3 = g_1^{r_B}$in group G_T, converti the data types of g1, g2, g3 into bit string;
5. Convert the data type of R_A and R_B into bit string and calculate $SK_B = KDF(ID_A \| ID_B \| R_A \| R_B \| g_1 \| g_2 \| g_3, klen)$;
6. (Optional) Calculate $S_B = Hash(0x82 \| g_1 \| Hash(g_2 \| g_3 \| ID_A \| ID_B \| R_A \| R_B))$;
7. Send R_B, (Optional S_B) to User A.

User A:

1. Verify whethe $RB \in G_1$ is established, and if not, the negotiation fails; otherwise, compute the elements $g_1^{`} = e(P_{pub-e}, P_2)^{r_A}, g_2^{`} = e(R_B, de_A), g_3^{`} = (g_2^{`})$, converting the data types of $g_1^{`}, g_2^{`}, g_3^{`}$ into a bit string;
2. Convert the data types of R_A and R_B into bit string, (optional) Calculate $S_1 = Hash(0x82 \| g_1^{`} \| Hash(g_2` \| g_3` \| ID_A \| ID_B \| R_A \| R_B))$, and checks whether $S_1 = S_B$ holds, if the equation is not established, the key confirmation from B to A fails.
3. Calculate $SK_A = KDF(ID_A \| ID_B \| R_A \| R_B \| g_1^{`} \| g_2^{`} \| g_3^{`}, klen)$;
4. (Optional) Calculate

$S_A = Hash(0x83 \| g_1^{'} \| Hash(g_2{'} \| g_3{'} \| ID_A \| ID_B \| R_A \| R_B))$and send the S_A to user B.

User B:

1. (Optional) $S_2 = Hash(0x83 \| g_1 \| Hash(g_2 \| g_3 \| ID_A \| ID_B \| R_A \| R_B))$is calculated and it is checked whether $S_2 = S_A$ is established. The key confirmation from A to B failed.

SM9 Key Encapsulation Mechanism and Public-key Cryptography

System Paremeters

The system parameter includes the curve identifier cid; the parameters of the elliptic curve base F_q; the parameters of the elliptic curve a and b ; the parameter of the twisting curve β (if the lower 4 bits of cid are 2); the prime factor of the curve N and the cofactor of N cf; the embedding times of the curve $E(F_q)$ with respect to N k; generation element P_1 of the N-th order cyclic subgroup \$\mathcal{G} $1 of$ *E(F*{qd1})$($d1$divides$k $); generationelement$P_2$ofthe$N$-thordercyclicsubgroup$\mathcal{G}$2of$ *E(F*{qd2})$($d2$divides$k$); theidentifierofthebilinearpairing$eeid; $(Option)$\psi $isthehomomorphicmapping$\mathcal{G}_1to\mathcal{G}_2\$.

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

System Signature Master Key and User Signature Key Generation

KGC generates the random key $ke \in [1, N-1]$ as the master private key, and calculates $P_{pub-e} = [ke]P_1$ in \$\mathcal{G} $1 as the master public key. The master key pair is$ *(ke, P*{pub-e})$and KGC keeps$ke$secret and$P_{pub-e}\$ public.

KGC chooses and exposes the encrypted private key represented by one byte to generate the function identifier hid.

The		hid, N) + ke$on the finite field$F_N.Ift_1 = 0

identity of user B is ID_B to generate the encrypted private key d_{eB} of user B, KGC first calculates $t_1 = H_1 (ID_B		, *itisnecessarytoregeneratethemasterprivatekey*, *comput* t_2 = ke · t_1^{-1}*andthencalculated*d_{eB} = [t_2] P_2$.

Key Encapsulation Algorithm

In order to encapsulate a key with a bit length $klen$ to user B, user A who is an encapsulator, needs to perform the following operation steps:

1.

Calculate the element $Q_B = [H_1 (ID_B		hid, N)] P_1 + P_{pub-e}*ingroup* \mathcal{G}_1$;

2. generate random numbers $r \in [1, N-1]$;
3. Calculate element $C = [r]Q_B$ in group $\mathcal{G}_1$, convert the data C into a bit string;
4. Calculate the element $g = e(P_{pub-e}, P_2)$ in group $\mathcal{G}_T$;
5. Calculate the element $w = g^r$ in group $\mathcal{G}_T$and convert the data w into a bit string.
6. Calculate $K = KDF(C \| w \| ID_B, klen)$, If K is an all 0 bit string, return step 2.
7. Output (K, C), where K is the key being encapsulated and C is the encapsulated ciphertext.

Key Decapsulation Algorithm

After user B receives the encapsulated ciphertext C, in order to decapsulate the key with the $klen$ bit length, the following steps need to be performed:

1. Verify that $C \in \mathcal{G}_1$ is valid, and if not, an error is reported and exit;

2. Calculate the element $w' = e(C, de_B)$ in the group $\mathcal{G}_T$ and convert the data type of w' into a bit string;
3. convert the data type of C into a bit string, and calculate the encapsulated key $K` = KDF(C\|w'\|ID_B, klen)$, if K' is all 0 bits String, the error and exit;
4. output key K'.

Public Key Encryption Algorithm

Suppose the message to be sent is the bit string M, $mlen$ is the bit length of M, K_1_len is the bit length of the key K_1 in the block cipher algorithm, and K_2_len is the bit length of the key K_2 in the function $MAC(K_2, Z)$.

To encrypt plaintext M to user B, user A as an encryptor, should implement the following computational steps:

1.

Calculate element \$Q_B = [H_1 (ID_B		hid, N)] P_1 + P_{pub-e}*ingroup* \mathcal{G}_1\$;

2. generate random numbers $r \in [1, N-1]$;
3. Calculate element $C_1 = [r]Q_B$ in group $\mathcal{G}_1$, convert the data C_1 into a bit string;
4. Calculate the element $g = e(P_{pub-e}, P_2)$ in group $\mathcal{G}_T$;
5. Calculate the element $w = g^r$ in group $\mathcal{G}_T$, convert the data w into a bit string;
6. Calculated by the method of encrypting plaintext:
7. If the method of encrypting a plaintext is based on a sequence cipher derived from a key-derived function, then 1. Calculate the integer klen = mlen + K_2_len and then calculate $K = KDF(C_1\|w\|ID_B, klen)$. Let K_1 be the leftmost $mlen$ bits of K, K_2 be the remaining K_2_len bits. If K_1 is a full 0-bit string, return to step 2; 2. Calculate $C_2 = M \oplus K_1$.
8. If the method of encrypting plaintext is a block cipher algorithm that combines key-derived functions, then 1. Calculate the integer klen = K_1_len + K_2_len and then calculate $K = KDF(C_1\|w\|ID_B, klen)$. Let K1 be the leftmost K_1_len bit of K, K_2 be the remaining K_2_len bits. If K_1 is a full 0-bit string, return to step 2; 2. Calculate $C_2 = Enc(K_1, M)$.
9. Calculate $C_3 = MAC(K_2, C_2)$;

10.

Output ciphertext \$C = C_1		C_3		C_2\$.

Public Key Decryption Algorithm

Let $mlen$ be the bit length of C_2 in the ciphertext $C = C_1 \| C_3 \| C_2$, $\boxed{\text{K\_1\_len}}$ is the bit length of the key K_1 in the block cipher algorithm, and $\boxed{\text{K\_2\_len}}$ is the bit length of the key K_2 in the function $MAC(K_2, Z)$.

In order to decrypt C, user B as a decryptor should implement the following computational steps:

1. Extract the bit string C_1 from C, convert the data C_1 into a point on the elliptic curve and verify whether $C_1 \in \mathcal{G}_1$, if not, report an error and exit;
2. Calculate the element $w' = e(C_1, de_B)$ in the group $\mathcal{G}_T$ and convert the data w' into a bit string;
3. Calculated by the method of encrypting plaintext:
4. If the method of encrypting a plaintext is based on a sequence cipher derived from a key-derived function 1. Calculate the integer $\boxed{\text{klen = mlen + K\_2\_len}}$ and then calculate $K' = KDF(C_1 \| w' \| ID_B, klen)$. Let K_1' be the leftmost $mlen$ bits of K', K_2 be the remaining $\boxed{\text{K\_2\_len}}$ bits. If K_1 is a full 0-bit string, then an error is reported and exit; 2. Calculate $M' = C2 \oplus K_1'$.
5. If the method of encrypting plaintext is a block cipher algorithm that combines key-derived functions, then 1. Calculate the integer $\boxed{\text{klen = K\_1\_len + K\_2\_len}}$ and then calculate $K = KDF(C_1 \| w \| ID_B, klen)$. Let K1 be the leftmost $\boxed{\text{K\_1\_len}}$ bit of K, K_2 be the remaining $\boxed{\text{K\_2\_len}}$ bits. If K_1 is a full 0-bit string, then an error is reported and exit; 2. Calculate $M' = Dec(K_1', C_2)$.
6. Calculate $u = MAC(K_2', C_2)$, extract the bit string C_3 from C, and if $u \in C_3$, then error and exit;
7. Output plain text M'.

The GmSSL Project is maintained by Zhi Guan.