The SM9 Cryptographic Schemes

Zhaohui Cheng

Independent Consultant
zhaohui_cheng@hotmail.com

Abstract. SM9 is a Chinese official cryptography standard which defines a set of identity-based cryptographic schemes from pairings. This report describes the technical specification of SM9. The security of schemes is also analyzed.

1 Introduction

In this document, the identity-based signature (IBS), the identity-based key agreement(IB-KA) and the identity-based encryption (IBE) scheme from SM9 [15] are described. These schemes are instantiated with an efficient bilinear pairing on elliptic curves [18] such as the optimal Ate pairing [26] or the R-Ate pairing [21].

Without loss of generality, a pairing is defined as a bilinear map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ are additive groups and $\mathbb{G}_T$ is a multiplicative group. All three groups have prime order r.

The map $\hat{e}$ has the following properties:

1. Bilinearity. For all $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$ and all $a, b \in \mathbb{Z}$, $\hat{e}([a]P, [b]Q) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy. For generator $P_1 \in \mathbb{G}_1$ and $P_2 \in \mathbb{G}_2$, $\hat{e}(P_1, P_2) \neq 1$

The report is organized as follows. First, used notation is defined in Section 2 and two supporting functions are described in Section 3. Then, three identity-based cryptographic schemes including signature, key agreement and encryption from the SM9 standard are presented in Section 4, 5 and 6 respectively. In Section 7, the security of the schemes is analyzed. Finally, performance comparison between SM9 and standardized schemes is given in Section 8.

2 Notation

The following list briefly describes the notation used in the document. One may refer to ISO/IEC 18033-2 [19] for detailed definitions.

1. $BITS(m)$: the primitive to count bit length of a bit string m.
2. $BS2IP(m)$: the primitive to convert a bit string m to an integer.

3. $EC2OSP(C)$: the primitive to convert an elliptic curve point C to an octet string.
4. $FE2OSP(w)$: the primitive to convert a field element w to an octet string.
5. $I2OSP(m, \ell)$: the primitive to convert an integer m to an octet string of length ℓ.

3 Supporting Functions

Before presenting the main schemes, two supporting functions used in the schemes are described here.

The first function is a key derivation function (KDF) which works as KDF2 in ISO/IEC 18033-2 [19].

KDF2 (H_v, Z, ℓ). Given a hash function H_v with v-bit output, a bit string Z and a non-negative integer ℓ

1. Set a 32-bit counter ct = 0x00000001.
2. For $i = 1$ to $\lceil \ell/v \rceil$.
   (a) Set $Ha_i = H_v(Z \| I2OSP(ct, 4))$.
   (b) Set $ct = ct + 1$.
3. Output the first ℓ bits of $Ha_1 \| Ha_2 \| \cdots \| Ha_{\lceil \ell/v \rceil}$.

The second function is a hash to range function (H2RF) which runs as follows: $\mathbf{H2RF}_i(H_v, Z, n)$. Given a hash function H_v with v-bit output, a bit string Z and a non-negative integer n and a non-negative integer index i

1. Set $\ell = 8 \times \lceil (5 \times BITS(n))/32 \rceil$.
2. Set $Ha =$**KDF2**$(H_v, I2OSP(i, 1) \| Z, \ell)$.
3. Set $h = BS2IP(Ha)$.
4. Output $h_i = (h \mod (n-1)) + 1$.

4 Identity-Based Signature

The SM9 signature (SM9-IBS) scheme consists of following four operations: **Setup**, **Private-Key-Extract**, **Sign** and **Verify**.

Setup $\mathbb{G}_{\mathtt{ID}}(1^k)$**.** On input the security parameter k, the operation runs as follows:

1. Generate three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order r and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Pick random generator $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$.
2. Pick a random $s \in \mathbb{Z}_r^*$ and compute $P_{pub} = [s]P_2$.
3. Set $g = \hat{e}(P_1, P_{pub})$.
4. Pick a cryptographic hash function H_v and a one byte appendix hid.
5. Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, P_{pub}, g, H_v, hid)$ and the master secret key $M_{\mathtt{sk}} = s$. SM9 standard requires $hid = 1$.

Private-Key-Extract $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$**.** Given an identity string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the operation outputs error if

$$s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r) \mod r = 0,$$

otherwise outputs

$$D_A = [\frac{s}{s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)}]P_1.$$

Sign$(M_{\mathfrak{pk}}, D_A, M)$. Given the message M, the private key D_A and the master public key $M_{\mathfrak{pk}}$, the operation runs as follows:

1. Pick a random $x \in \mathbb{Z}_r^*$.
2. Set $w = g^x$.
3. Set $h = \mathbf{H2RF}_2(H_v, M \| FE2OSP(w), r)$.
4. Set $l = (x - h) \mod r$.
5. Set $S = [l]D_A$.
6. Output $\langle h, S \rangle$.

Verify$(M_{\mathfrak{pk}}, \mathtt{ID}_A, M, \langle h, S \rangle)$. Given the message M, the signer's identity string $\mathtt{ID}_A$, the signature $\langle h, S \rangle$ and the master public key $M_{\mathfrak{pk}}$, the operation runs as follows:

1. If $h \notin \mathbb{Z}_r^*$ or $S \notin \mathbb{G}_1^*$, then output failure and terminate.
2. Set $h_1 = \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)$.
3. Set $Q = [h_1]P_2 + P_{pub}$.
4. Set $u = \hat{e}(S, Q)$.
5. Set $t = g^h$.
6. Set $w' = u \cdot t$.
7. Set $h_2 = \mathbf{H2RF}_2(H_v, M \| FE2OSP(w'), r)$.
8. If $h \neq h_2$, then output failure, otherwise output success.

5 Identity-Based Key Agreement

The SM9 key agreement (SM9-KA) is an authenticated two-pass (or three-pass) key agreement (with key confirmation). The scheme consists of following operations: **Setup**, **Private-Key-Extract**, **Message Exchange**, **Session Key Generation** and **Session Key Confirmation**.

Setup $\mathbb{G}_{\mathtt{ID}}(1^k)$**.** On input security parameter k, the operation runs as follows:

1. Generate three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order r and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Pick random generator $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$.
2. Pick a random $s \in \mathbb{Z}_r^*$ and compute $P_{pub} = [s]P_1$.
3. Set $g = \hat{e}(P_{pub}, P_2)$.
4. Pick a cryptographic hash function H_v and a one byte appendix hid.
5. Output the master public key $M_{\mathfrak{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, P_{pub}, g, H_v, hid)$ and the master secret key $M_{\mathfrak{sk}} = s$. SM9 standard requires $hid = 3$.

Private-Key-Extract $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$**.** Given an identity string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathfrak{pk}}$ and $M_{\mathfrak{sk}}$, the operation outputs error if

$$s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r) \mod r = 0,$$

otherwise outputs

$$D_A = [\frac{s}{s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)}]P_2.$$

Message Exchange.

$$\begin{array}{l} A \to B : R_A = [x_A]([\mathbf{H2RF}_1(H_v, \mathtt{ID}_B \| hid, r)]P_1 + P_{pub}) \\ B \to A : R_B = [x_B]([\mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)]P_1 + P_{pub}), S_B \\ A \to B : S_A \end{array}$$

where random $x_A, x_B \in \mathbb{Z}_r^*$ are picked by A and B respectively and S_B and S_A are the optional session key confirmation parts. The method to generate such optional values is explained later.

Session Key Generation.

1. Entity A computes key values

$$g_1 = \hat{e}(R_B, D_A), g_2 = \hat{e}(P_{pub}, P_2)^{x_A} = g^{x_A}, g_3 = g_1^{x_A}.$$

2. Entity A computes ℓ-bit session key

$$SK_A = \mathbf{KDF2}(H_v, \mathtt{ID}_A \| \mathtt{ID}_B \| EC2OSP(R_A) \| EC2OSP(R_B) \| FE2OSP(g_1) \| FE2OSP(g_2) \| FE2OSP(g_3), \ell).$$

3. Entity B computes key values

$$g_1' = \hat{e}(P_{pub}, P_2)^{x_B} = g^{x_B}, g_2' = \hat{e}(R_A, D_B), g_3' = {g_2'}^{x_B}.$$

4. Entity B computes ℓ-bit session key

$$SK_B = \mathbf{KDF2}(H_v, \mathtt{ID}_A \| \mathtt{ID}_B \| EC2OSP(R_A) \| EC2OSP(R_B) \| FE2OSP(g_1') \| FE2OSP(g_2') \| FE2OSP(g_3'), \ell).$$

Session Key Confirmation.

1. Entity B computes its key confirmation

$$S_B = H_v(0x82 \| FE2OSP(g_2') \| H_v(FE2OSP(g_1') \| FE2OSP(g_3') \| \mathtt{ID}_A \| \mathtt{ID}_B \| EC2OSP(R_A) \| EC2OSP(R_B)).$$

   Entity A should verify S_B's correctness with g_1, g_2, g_3.
2. Entity A computes its key confirmation

$$S_A = H_v(0x83 \| FE2OSP(g_1) \| H_v(FE2OSP(g_2) \| FE2OSP(g_3) \| \mathtt{ID}_A \| \mathtt{ID}_B \| EC2OSP(R_A) \| EC2OSP(R_B)).$$

   Entity B should verify S_A's correctness with g_1', g_2', g_3'.

Note that entity $A(B)$ should check $R_B(R_A)$ lies in $\mathbb{G}_1^*$ respectively.

6 Identity-Based Encryption

The SM9 encryption (SM9-IBE) is a hybrid encryption scheme built from an identity-based key encapsulation scheme (KEM) and a data encapsulation scheme (DEM). DEM can be one of those schemes such as DEM2 or DEM3 standardized in ISO/IEC 18033-2 [19]. First, the SM9 KEM is presented, then the hybrid encryption scheme is described. SM9-KEM scheme consists of four operations: **Setup**, **Private-Key-Extract**, **KEM-Encap** and **KEM-Decap** as follows:

Setup $\mathbb{G}_{\mathtt{ID-KEM}}(1^k)$**.** On input security parameter k, the operation runs as follows:

1. Generate three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order r and a bilinear pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Pick random generator $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2$.
2. Pick a random $s \in \mathbb{Z}_r^*$ and compute $P_{pub} = [s]P_1$.
3. Set $g = \hat{e}(P_{pub}, P_2)$.
4. Pick a cryptographic hash function H_v and a one byte appendix hid.
5. Output the master public key $M_{\mathtt{pk}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, P_{pub}, g, H_v, hid)$ and the master secret key $M_{\mathtt{sk}} = s$. SM9 standard requires $hid = 3$.

Private-Key-Extract $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, M_{\mathtt{sk}}, \mathtt{ID}_A)$**.** Given an identity string $\mathtt{ID}_A \in \{0,1\}^*$ of entity A, $M_{\mathtt{pk}}$ and $M_{\mathtt{sk}}$, the operation outputs error if

$$s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r) \mod r = 0,$$

otherwise outputs

$$D_A = [\frac{s}{s + \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)}]P_2.$$

KEM-Encap $\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A)$. Given an identify string $\mathtt{ID}_A$ and the master public key $M_{\mathtt{pk}}$, the operation runs as follows:

1. Set $h_1 = \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)$.
2. Set $Q = [h_1]P_1 + P_{pub}$.
3. Pick a random $x \in \mathbb{Z}_r^*$.
4. Set $C_1 = [x]Q$.
5. Set $t = g^x$.
6. Set $K = \mathbf{KDF2}(H_v, EC2OSP(C_1) \| FE2OSP(t) \| \mathtt{ID}_A, \ell)$, where ℓ is the key length of the DEM.
7. Output $\langle K, C_1 \rangle$.

KEM-Decap $\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C_1)$. Given an identify string $\mathtt{ID}_A$, the corresponding private key D_A, the encapsulation part C_1 and the master public key $M_{\mathtt{pk}}$, the operation runs as follows:

1. If $C_1 \notin \mathbb{G}_1^*$, then output $\perp$ and terminate.
2. Set $t = \hat{e}(C_1, D_A)$.
3. Set $K = \mathbf{KDF2}(H_v, EC2OSP(C_1) \| FE2OSP(t) \| \mathtt{ID}_A, \ell)$, where ℓ is the key length of the DEM.

4. Output K.

The full SM9 encryption scheme using DEM3 as specified in [19] works as follows:

KEM-DEM-Encrypt $\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, m)$. Given an identify string $\mathtt{ID}_A$, the plain text m and the master public key $M_{\mathfrak{pk}}$, the operation runs as follows:

1. Set $h_1 = \mathbf{H2RF}_1(H_v, \mathtt{ID}_A \| hid, r)$.
2. Set $Q = [h_1]P_1 + P_{pub}$.
3. Pick a random $x \in \mathbb{Z}_r^*$.
4. Set $C_1 = [x]Q$.
5. Set $t = g^x$.
6. Set $K_1 \| K_2 = \mathbf{KDF2}(H_v, EC2OSP(C_1) \| FE2OSP(t) \| \mathtt{ID}_A, BITS(m) + v)$.
7. Set $C_2 = K_1 \oplus m$.
8. Set $C_3 = H_v(C_2 \| K_2)$.
9. Output $\langle C_1, C_2, C_3 \rangle$.

KEM-DEM-Decrypt $\mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A, \langle C_1, C_2, C_3 \rangle)$. Given an identify string $\mathtt{ID}_A$, the corresponding private key D_A, the cipher text $\langle C_1, C_2, C_3 \rangle$ and the master public key $M_{\mathfrak{pk}}$, the operation runs as follows:

1. If $C_1 \notin \mathbb{G}_1^*$, then output $\perp$ and terminate.
2. Set $t = \hat{e}(C_1, D_A)$.
3. Set $K_1 \| K_2 = \mathbf{KDF2}(H_v, EC2OSP(C_1) \| FE2OSP(t) \| \mathtt{ID}_A, BITS(C_2) + v)$.
4. Set $C_3' = H_v(C_2 \| K_2)$.
5. If $C_3' \neq C_3$, then output $\perp$ and terminate.
6. Output $m = K_1 \oplus C_2$.

7 Security Analysis of The Schemes

In this section, we analyze the security of the schemes specified in the SM9 standard. The security analysis of SM9-IBS is very similar to the one given in [1]. In particular, as Theorem 1 in [1], a security reduction of SM9-IBS can be constructed based on the τ-SDH complexity assumption defined below. Hence, we skip the details of this reduction. Instead, we focus on the security analysis of SM9-KA and SM9-IBE.

7.1 Related Complexity Assumptions

Before giving the detailed security analysis, we present some related complexity assumptions. We follow the approach in [10] to use $i \in \{1, 2\}$ to denote different choices of $P_i \in \mathbb{G}_i$ and an assumption with subscripts such as $\mathrm{BDH}_{i,j,k}$ identifies how the three elements are chosen from the groups for the assumption. Symbol $\in_R$ denotes randomly sampling from a set.

Assumption 1 (τ-Strong Diffie-Hellman (τ-SDH) [6, 24]) *For a positive integer τ and $\alpha \in_R \mathbb{Z}_r^*$, given $(P_i, [\alpha]P_i, [\alpha^2]P_i, \ldots, [\alpha^\tau]P_i)$ for some value $i \in \{1, 2\}$, computing $(h, [\frac{1}{\alpha+h}]P_i)$ for some $h \in \mathbb{Z}_r^*$ is hard.*

Boneh and Boyen [6] proved a lower bound on the computational complexity of the τ-SDH problem in the generic group model [25]: By assuming that the best discrete logarithm (DL) algorithm on the chosen groups is the general DL algorithm, at least $\sqrt{r/\tau}$ group operations are required to solve the τ-SDH problem when $\tau < o(\sqrt[3]{r})$.

Assumption 2 (Bilinear Diffie-Hellman (BDH) [7]) *For $a, b, c \in_R \mathbb{Z}_r^*$, given $(P_1, P_2, [a]P_i, [b]P_j, [c]P_k)$, for some values of $i, j, k \in \{1, 2\}$, computing $\hat{e}(P_1, P_2)^{abc}$ is hard.*

Assumption 3 (ψ-Bilinear Diffie-Hellman (ψ-BDH)) *For $a, b \in_R \mathbb{Z}_r^*$, given $(P_i, P_j, [a]P_j, [b]P_j)$ for some values of $i, j \in \{1, 2\}$ and $i \neq j$, computing $\hat{e}(P_1, P_2)^{ab}$ is hard if no group homomorphism $\psi: \mathbb{G}_j \to \mathbb{G}_i$ such that $\psi(P_j) = P_i$ is efficiently computable.*

ψ-BDH$_{i,j}$ is called BDH-j in [13] without explicit restriction on ψ. Apparently, Assumption 3 is not weaker than Assumption 2 because by calling a BDH algorithm with $(P_i, P_j, [a]P_j, [b]P_j, P_j)$, $\hat{e}(P_1, P_2)^{ab}$ is computed. However, it is subtle to decide the exact relationship between these two assumptions. Depending on the difficulty to compute $\psi : \mathbb{G}_j \to \mathbb{G}_i$, there are two cases.

- Case 1. For Type-1 and Type-2 pairings [14], an efficient group homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ exists. However, such ψ may not satisfy $\psi(P_2) = P_1$ if P_1 and P_2 are chosen randomly and independently. For example, with a Type-1 pairing and ψ as the identity map, two random generators may be chosen as follows: $P_2 = [c]P_1$ for some random $c \in \mathbb{Z}_r^*$. Similar situation could happen to the Type-2 pairing with ψ as the trace map, and groups are constructed as follows [10]. Let E be an elliptic curve defined over a prime field $\mathbb{F}_p$ with an embedding degree ν. Set $P_1 = \mathcal{P}_1$ and $P_2 = [\frac{1}{\nu}]\mathcal{P}_1' + \mathcal{P}_2$, where $\mathcal{P}_1, \mathcal{P}_1' \in E(\mathbb{F}_p)$ are two random generator of a cyclic group $\mathcal{G}_1$ with order r and $\mathcal{P}_2 \in E(\mathbb{F}_{p^\nu})$ is a generator of a cyclic group $\mathcal{G}_2$ with order r. $\psi(P_2) = \mathcal{P}_1' = [c]P_1$ for some $c \in \mathbb{Z}_r^*$. In this setting, ψ-BDH$_{1,2}$ equals to BDH$_{2,2,k}$. Given a BDH$_{2,2,2}$ problem $(P_1, P_2', [a]P_2', [b]P_2', [c]P_2')$ with $\psi(P_2') = P_1$ (note that in the Type-2 pairing $P_2' = [\frac{1}{\nu}]\mathcal{P}_1 + \mathcal{P}_2$), we can use a ψ-BDH$_{1,2}$ algorithm to solve the problem $(\psi([c]P_2'), P_2', [a]P_2', [b]P_2')$ to get $\hat{e}(\psi([c]P_2'), P_2')^{ab} = \hat{e}([c]P_1, P_2')^{ab} = \hat{e}(P_1, P_2')^{abc}$. Similarly, BDH$_{2,2,1}$ is solvable by ψ-BDH$_{1,2}$.
- Case 2. For Type-2 pairings, there appears no efficiently computable homomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$. For Type-3 pairings, there are no known efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ [14]. In both cases, there appears no simple way to solve a ψ-BDH problem (ψ-BDH$_{2,1}$ for Type-2 pairings) other than relying on an algorithm like the one for the BDH problem.

More on the role of group homomorphism ψ on cryptographic protocols employing asymmetric pairings can be found in [8].

Assumption 4 (τ-BDHI [5]) *For a positive integer τ and $\alpha \in_R \mathbb{Z}_r^*$, given ($P_1$, P_2, $[\alpha]P_i$, $[\alpha^2]P_i$, $\ldots$, $[\alpha^\tau]P_i$) for some value $i \in \{1,2\}$, computing $\hat{e}(P_1,P_2)^{1/\alpha}$ is hard.*

Assumption 5 (Bilinear Collision Attack Assumption (τ-BCAA1) [9]) *For a positive integer τ and $\alpha \in_R \mathbb{Z}_r^*$, given ($P_1$, P_2, $[\alpha]P_i, h_0, (h_1, [\frac{\alpha}{h_1+\alpha}]P_j), \ldots$, $(h_\tau, [\frac{\alpha}{h_\tau+\alpha}]P_j)$) for some values of $i,j \in \{1,2\}$ where $h_i \in_R \mathbb{Z}_r^*$ and different from each other for $0 \le i \le \tau$, computing $\hat{e}(P_1,P_2)^{\alpha/(h_0+\alpha)}$ is hard.*

Note that Assumption 5 is slight different from the one given in [9]. However, the following Lemma 1 together with Theorem 7 in [9] shows that Assumption 5 is equivalent to the one defined in [9].

Lemma 1 *If there exists a polynomial time algorithm to solve (τ-1)-BCAA1$_{i,2}$, then there exists a polynomial time algorithm for τ-BDHI$_2$, if there exists an efficient isomorphism ψ: $\mathbb{G}_2 \to \mathbb{G}_1$.*

Proof: If there is a polynomial time algorithm $\mathcal{A}$ to solve the (τ-1)-BCAA1$_{i,2}$ problem, we can construct a polynomial time algorithm $\mathcal{B}$ to solve the τ-BDHI$_2$ problem as follows. Given an instance of the τ-BDHI$_2$ problem

$$(P_1, P_2, [x]P_2, [x^2]P_2, \ldots, [x^\tau]P_2),$$

$\mathcal{B}$ works as follows to compute $\hat{e}(P_1,P_2)^{1/x}$.

1. Randomly choose different $h_0, \ldots, h_{\tau-1} \in \mathbb{Z}_r^*$. Let $f(z)$ be the polynomial
$$f(z) = \prod_{a=1}^{\tau-1}(z+h_a) = \sum_{a=0}^{\tau-1} c_a z^a.$$
The constant term c_0 is non-zero because h_a's are different and c_i is computable from h_a's.
2. Set
$$Q_2 = \sum_{a=0}^{\tau-1}[c_a x^a]P_2 = [f(x)]P_2,$$
and
$$[x]Q_2 = \sum_{a=0}^{\tau-1}[c_a x^{a+1}]P_2 = [xf(x)]P_2.$$
3. Set
$$f_b(z) = \frac{z-h_0}{z+h_b}f(z) = \sum_{a=0}^{\tau-1} d_a z^a,$$
and compute
$$[\frac{x-h_0}{x+h_b}]Q_2 = [\frac{x-h_0}{x+h_b}f(x)]P_2 = [f_b(x)]P_2 = \sum_{a=0}^{\tau-1}[d_a x^a]P_2$$
for $1 \le b \le \tau-1$.

4. Set $Q_1 = \psi(Q_2)$ and pass the following instance of the (τ-1)-BCAA1$_{i,2}$ problem to $\mathcal{A}$

$$(Q_1, Q_2, \psi([x-h_0]Q_2), h_0, (h_1+h_0, [\frac{x-h_0}{x+h_1}]Q_2), \ldots, (h_{\tau-1}+h_0, [\frac{x-h_0}{x+h_{\tau-1}}]Q_2))$$

if $i = 1$, or

$$(Q_1, Q_2, [x-h_0]Q_2, h_0, (h_1+h_0, [\frac{x-h_0}{x+h_1}]Q_2), \ldots, (h_{\tau-1}+h_0, [\frac{x-h_0}{x+h_{\tau-1}}]Q_2))$$

to get

$$T = \hat{e}(Q_1, Q_2)^{\frac{x-h_0}{x}} = \hat{e}(Q_1, Q_2) \cdot \hat{e}(Q_1, Q_2)^{-h_0/x}.$$

5. Note that

$$[\frac{1}{x}](Q_2 - [c_0]P_2) = [\frac{1}{x}]([f(x)]P_2 - [c_0]P_2) = \sum_{a=1}^{\tau-1}[c_a x^{a-1}]P_2.$$

Set

$$T' = \sum_{a=1}^{\tau-1}[c_a x^{a-1}]P_2 = [\frac{f(x)-c_0}{x}]P_2.$$

Then,

$$T_0 = \hat{e}(\psi(T'), Q_2 + [c_0]P_2) = \hat{e}([f(x)-c_0]P_1, Q_2 + [c_0]P_2)^{1/x} =$$

$$\hat{e}(Q_1, Q_2)^{1/x} \cdot \hat{e}(P_1, P_2)^{-c_0^2/x}.$$

Finally, compute

$$\hat{e}(P_1, P_2)^{1/x} = ((T/\hat{e}(Q_1, Q_2))^{-1/h_0}/T_0)^{1/c_0^2}.$$

□

Assumption 6 (Decision BIDH (DBIDH) [9]) *For $a, b, r \in_R \mathbb{Z}_r^*$, differentiating*

$$(P_1, P_2, [a]P_i, [b]P_j, \hat{e}(P_1, P_2)^{b/a}) \text{ and } (P_1, P_2, [a]P_i, [b]P_j, \hat{e}(P_1, P_2)^r),$$

for some values of $i, j \in \{1, 2\}$, is hard.

Assumption 7 (Gap-τ-BCAA1 [9]) *For a positive integer τ and $\alpha \in_R \mathbb{Z}_r^*$, given $(P_1, P_2, [\alpha]P_i, h_0, (h_1, [\frac{\alpha}{h_1+\alpha}]P_j), \ldots, (h_\tau, [\frac{\alpha}{h_\tau+\alpha}]P_j))$ for some values of $i, j \in \{1, 2\}$ where $h_i \in_R \mathbb{Z}_r^*$ and different from each other for $0 \leq i \leq \tau$, and an efficient algorithm of DBIDH, computing $\hat{e}(P_1, P_2)^{\alpha/(h_0+\alpha)}$ is hard.*

Note that by Lemma 1, Assumption 7 is equivalent to a Gap-τ-BDHI$_2$ assumption that is given the access to a DBIDH oracle and an efficient ψ, τ-BDHI$_2$ problem is still hard. The relationship among these assumptions can be found in [9].

7.2 Security Analysis of SM9-KA

We use a modified Bellare-Rogaway key exchange model [2, 4, 10] to analyse the two-pass SM9-KA. In the model, each party participating in a session is treated as an oracle. An oracle $\Pi_{i,j}^s$ denotes the s-th instance of party i involved with a partner party j in a session. The oracle $\Pi_{i,j}^s$ executes the prescribed protocol Π and produces the output as $\Pi(1^k, i, j, SK_i, PK_i, PK_j, tran_{i,j}^s, r_{i,j}^s, x)$=$(m, \delta_{i,j}^s, \sigma_{i,j}^s, j)$ where $r_{i,j}^s$ is the random flips of the oracle; x is the input message; m is the outgoing message; SK_i and PK_i are the private/public key pair of party i; $\delta_{i,j}^s$ is the decision of the oracle (accept/reject the session or no decision yet); $\sigma_{i,j}^s$ is the generated session key and PK_j is the public key of the intended partner j (see [2, 4, 10] for more details). After the response is generated, the conversation transcript $tran_{i,j}^s$ is updated as $tran_{i,j}^s.x.m$, where "$a.b$" denotes the result of the concatenation of two strings, a and b. An adversary can access an oracle by issuing some specified queries defined in the game below.

The security of a protocol is defined through a two-phase game between an adversary $\mathcal{A}$ and a simulator which simulates the executions of a protocol by providing the adversary with access to oracles. In the first phase, $\mathcal{A}$ is allowed to issue the following queries to oracles in any order.

1. $Send(\Pi_{i,j}^s, x)$. Upon receiving the message x, oracle $\Pi_{i,j}^s$ executes the protocol and responds with an outgoing message m or a decision to indicate accepting or rejecting the session. If the oracle $\Pi_{i,j}^s$ does not exist, it will be created as an initiator, the party who sends out the first message in the protocol, if $x = \lambda$, or as a responder otherwise. In this report, we require $i \neq j$, namely, a party will not run a session with itself. Such restriction is not unusual in practice.
2. $Reveal(\Pi_{i,j}^s)$. If the oracle has not accepted, it returns $\perp$; otherwise, it reveals the session key.
3. $Corrupt(i)$. The party i responds with its private key.

Once the adversary decides that the first phase is over, it starts the second phase by choosing a *fresh oracle* $\Pi_{i,j}^s$ and issuing a $Test(\Pi_{i,j}^s)$ query, where the *fresh oracle* $\Pi_{i,j}^s$ and $Test(\Pi_{i,j}^s)$ query are defined as follows.

Definition 1 (fresh oracle) *An oracle $\Pi_{i,j}^s$ is fresh if (1) $\Pi_{i,j}^s$ has accepted; (2) $\Pi_{i,j}^s$ is unopened (not been issued the* Reveal *query); (3) party $j \neq i$ is not corrupted (not been issued the* Corrupt *query); (4) there is no opened oracle $\Pi_{j,i}^t$, which has had a matching conversation to $\Pi_{i,j}^s$.*

The above fresh oracle is particularly defined to cover the key-compromise impersonation resilience property since it implies that party i could have been issued a *Corrupt* query.

4. $Test(\Pi_{i,j}^s)$. Oracle $\Pi_{i,j}^s$ which is fresh, as a challenger, randomly chooses $b \in \{0, 1\}$ and responds with the session key if $b = 0$, or a random sample from the distribution of the session key otherwise.

After this point the adversary can continue querying the oracles except that it cannot reveal the test oracle $\Pi^s_{i,j}$ or an oracle $\Pi^t_{j,i}$ which has a matching conversation to $\Pi^s_{i,j}$ if such an oracle exists, and it cannot corrupt party j. Finally the adversary outputs a guess b' for b. If $b' = b$, we say that the adversary wins. The adversary's advantage is defined as

$$\mathtt{Adv}_{\mathcal{A}}(k) = |2\Pr[b' = b] - 1|.$$

We use session ID to define *matching conversations*. Two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have a matching conversation to each other if both of them have the same session ID. In this report, we will use the concatenation of the messages in a session (the transcript of an oracle) to define the session ID.

A secure authenticated key (AK) agreement protocol is defined as follows.

Definition 2 *Protocol Π is a secure AK if:*

1. *In the presence of a benign adversary, which faithfully conveys messages, on $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$, both oracles always accept holding the same session key, and this key is distributed uniformly in the session key space;*
2. *For any polynomial time adversary $\mathcal{A}$, $\mathtt{Adv}_{\mathcal{A}}(k)$ is a negligible function of security parameter k.*

If a protocol is secure regarding the above formulation, it achieves implicit mutual key authentication and the following general security properties: the known session key security, the key-compromise impersonation resilience and the unknown key-share resilience [4, 10].

Now we consider the forward secrecy property. Informally, the forward secrecy of a protocol requires that the security of a session key established by a party is not affected even if the long-term key of either the party is compromised afterwards.

Definition 3 *An AK protocol is said to be forward secure if any polynomial time adversary wins the game with negligible advantage when it chooses as the challenger (i.e. in place of the fresh oracle) an* unopened *oracle $\Pi^s_{i,j}$ which has a matching conversation to another* unopened *oracle $\Pi^t_{j,i}$ and both oracles accepted and only one of i and j can be corrupted. If both i and j can be corrupted, then the protocol achieves perfect forward secrecy. If in the game, the master secret key can be disclosed, then the protocol achieves master secret forward secrecy. The corruption of long-term keys or the disclosure of the master secret key may happen at any time of the game.*

Theorem 1 *SM9-KA is a secure AK, provided that* $\mathbf{H2RF}_1, \mathbf{KDF2}$ *are random oracles and the Gap-τ-BCAA1$_{1,2}$ assumption is sound. Specifically, suppose that there is an adversary $\mathcal{A}$ against the protocol with non-negligible probability $\epsilon(k)$ and running time $t(k)$, and in the attack* $\mathbf{H2RF}_1$ *has been queried $q_1 + 1$ times, and* $\mathbf{KDF2}$ *has been queried q_2 times, and q_o oracles have been created.*

Then there exists an algorithm $\mathcal{B}$ solving the Gap-q_1-BCAA1$_{1,2}$ problem with advantage

$$\mathtt{Adv}_{\mathcal{B}}(k) \geq \frac{\epsilon(k)}{(q_1+1)\cdot q_o}$$

within a running time

$$t_{\mathcal{B}} \leq t(k) + O(q_2 \cdot q_o \cdot \mathcal{O}),$$

where $\mathcal{O}$ is the time of one access to the DBIDH$_{1,1}$ oracle.

Proof: Condition 1 of Definition 2 directly follows from the protocol specification. In the sequel we prove that the protocol satisfies Condition 2. We show that if $\mathcal{A}$ exists, we can construct a probabilistic polynomial time (PPT) algorithm $\mathcal{B}$ to solve a Gap-q_1-BCAA1$_{1,2}$ problem with non-negligible probability.

Given an instance of the Gap-q_1-BCAA1$_{1,2}$ problem $(P_1, P_2, [x]P_1,\ h_0,\ (h_1,\ [\frac{x}{h_1+x}]P_2),\ \ldots, (h_{q_1},\ [\frac{x}{h_{q_1}+x}]P_2))$ with a set of pairing parameter where $h_i \in_R \mathbb{Z}_r^*$ for $0 \leq i \leq q_1$ and the DBIDH$_{1,1}$ oracle $\mathcal{O}_{DBIDH}$, $\mathcal{B}$ simulates $\mathbb{G}_{\mathtt{ID}}$ to generate the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T,\ \hat{e},\ P_1,\ P_2,\ [x]P_1,\ \hat{e}([x]P_1, P_2),\ H_v,\ hid)$, i.e., using x as the master secret key, which it does not know. Function $\mathbf{H2RF}_1$ and $\mathbf{KDF2}$ are constructed from the hash function H_v and are simulated as two random oracles controlled by $\mathcal{B}$.

We slightly abuse the notation $\Pi_{i,j}^t$ to refer to the t-th party instance among all the party instances created in the attack, instead of the t-th instance of party i. This would not affect the soundness of the security model.

$\mathcal{B}$ randomly chooses $1 \leq I \leq q_1 + 1$ and $1 \leq J \leq q_o$, and interacts with $\mathcal{A}$ in the following way:

- $\mathbf{H2RF}_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list $\mathbf{H2RF}_1^{list}$ of tuples $(\mathtt{ID}_i, h_i, D_i)$ as explained below. When $\mathcal{A}$ queries the oracle $\mathbf{H2RF}_1$ on $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ is on $\mathbf{H2RF}_1^{list}$ in a tuple $(\mathtt{ID}_i, h_i, D_i)$, then $\mathcal{B}$ responds with $\mathbf{H2RF}_1(\mathtt{ID}_i) = h_i$.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $\mathbf{H2RF}_1(\mathtt{ID}_I) = h_0$.
  - Otherwise, $\mathcal{B}$ selects a random integer h_i with $i > 0$ from the Gap-q_1-BCAA1$_{1,2}$ instance which has not been chosen by $\mathcal{B}$ and stores $(\mathtt{ID}_i,\ h_i,\ [\frac{x}{h_i+x}]P_2)$ into the tuple list. $\mathcal{B}$ responds with $\mathbf{H2RF}_1(\mathtt{ID}_i) = h_i$.
- $\mathbf{KDF2}(\mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t})$: $\mathcal{B}$ maintains a list $\mathbf{KDF2}^{list}$ of pairs in the form $(\langle \mathtt{ID}_i,\ \mathtt{ID}_j, R_i, R_j, g_{1t},\ g_{2t},\ g_{3t}\rangle,\ \zeta_t)$. To respond to a query, $\mathcal{B}$ does the following operations:
  - If $\mathbf{KDF2}^{list}$ has a tuple indexed by $\langle \mathtt{ID}_i,\ \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t}\rangle$, then $\mathcal{B}$ responds with ζ_t.
  - Otherwise, $\mathcal{B}$ goes through the list Λ built in the *Reveal* query to find tuples of the form $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j\rangle, r^t, \zeta^t, O^t)$ indexed by $\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j\rangle$ and proceeds as follows:
    - Set $R = R_j$ and $T = g_{1t}$ if $O^t = 1$, otherwise $R = R_i$ and $T = g_{2t}$, query $\mathcal{O}_{DBIDH}$ with $([x]P_1, P_2, [h_0 + x]P_1, R, T)$.

* If $\mathcal{O}_{DBIDH}$ returns 1 and $g_{3t} = T^{r^t}$, $\mathcal{B}$ removes the tuple from Λ and inserts $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t}\rangle, \zeta^t)$ into $\mathbf{KDF2}^{list}$, and finally responds with ζ^t.

- Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_t \in \{0,1\}^\ell$ and inserts a new tuple $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t}\rangle, \zeta_t)$ into the list $\mathbf{KDF2}^{list}$. It responds to $\mathcal{A}$ with ζ_t.

– Corrupt($\mathtt{ID}_i$): $\mathcal{B}$ looks through list $\mathbf{H2RF}_1^{list}$. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $\mathbf{H2RF}_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of D_i: if $D_i \neq \perp$, then $\mathcal{B}$ responds with D_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

– Send($\Pi_{i,j}^t, R$): $\mathcal{B}$ maintains a list with tuples of $(\Pi_{i,j}^t, r_{i,j}^t, tran_{i,j}^t)$ and responds to the query as follows:

- If $t \neq J$, $\mathcal{B}$ randomly chooses $r^t \in \mathbb{Z}_r^*$ as the random flips of the oracle and generates $[r^t]([\mathbf{H2RF}_1(\mathtt{ID}_j)]P_1 + [x]P_1)$ as the message.
- If $t = J$, $\mathcal{B}$ further checks the value of D_j corresponding $\mathtt{ID}_j$ on the list $\mathbf{H2RF}_1^{list}$ after querying $\mathbf{H2RF}_1(\mathtt{ID}_j)$, and then responds the query differently as below depending on this value.
  * If $D_j \neq \perp$, $\mathcal{B}$ aborts the game (**Event 2**). We note that there is only one party's private key is represented as $\perp$ in the whole simulation.
  * Otherwise, $\mathcal{B}$ randomly chooses $y \in \mathbb{Z}_r^*$ and responds with $[y]P_1$. Note that $\Pi_{i,j}^t$ can be the initiator (if $R = \lambda$) or the responder (if $R \neq \lambda$), and $\mathcal{B}$ doesn't know the random flips of the oracle.

– Reveal($\Pi_{i,j}^t$): $\mathcal{B}$ maintains a list Λ with tuples $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j\rangle, r^t, \zeta^t, O^t)$. $\mathcal{B}$ responds to the query as follows:

- If $t = J$ or if the J-th oracle has been generated as $\Pi_{a,b}^J$ and $\mathtt{ID}_a = \mathtt{ID}_j, \mathtt{ID}_b = \mathtt{ID}_i$ and two oracles have the same session ID, then abort the game (**Event 3**).
- Go through $H_1^{list}(\mathtt{ID}_i)$ to find the private key D_i of party i with identity $\mathtt{ID}_i$.
- If $D_i \neq \perp$, compute $g_1 = \hat{e}(R_j, D_i), g_2 = \hat{e}([x]P_1, P_2)^{r^t}, g_3 = g_1^{r^t}$ where R_j is the incoming message and r^t is the random flips of the oracle $\Pi_{i,j}^t$. $\mathcal{B}$ responds with $\mathbf{KDF2}(\mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_1, g_2, g_3)$ if the oracle is the initiator, or $\mathbf{KDF2}(\mathtt{ID}_j, \mathtt{ID}_i, R_j, R_i, g_2, g_1, g_3)$ otherwise.
- Otherwise, go through $\mathbf{KDF2}^{list}$ to find tuples indexed by $\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, *, \hat{e}([x]P_1, P_2)^{r^t}, *\rangle$ (if $\Pi_{i,j}^t$ is the initiator) or by $\langle \mathtt{ID}_j, \mathtt{ID}_i, R_j, R_i, \hat{e}([x]P_1, P_2)^{r^t}, *, *\rangle$ (if $\Pi_{i,j}^t$ is the responder). $*$ matches any values. For each $(g_{1t}, g_{2t}, g_{3t}, \zeta_t)$ in the found tuples,
  * Set $R = R_j$ and $T = g_{1t}$ if $\Pi_{i,j}^t$ is the initiator, otherwise $R = R_i$ and $T = g_{2t}$; query $\mathcal{O}_{DBIDH}$ with $([x]P_1, P_2, [h_0 + x]P_1, R, T)$.
  * If $\mathcal{O}_{DBIDH}$ returns 1 and $g_{3t} = T^{r^t}$, then $\mathcal{B}$ responds to the query with ζ_t.
- Otherwise (no match is found in the last step), randomly choose $\zeta^t \in \{0,1\}^\ell$ and insert $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j\rangle, r^t, \zeta^t, 1)$ if the oracle is the initiator or $(\langle \mathtt{ID}_j, \mathtt{ID}_i, R_j, R_i\rangle, r^t, \zeta^t, 0)$ into Λ. $\mathcal{B}$ responds with ζ^t.

– Test($\Pi^t_{i,j}$): If $t \neq J$ or ($t = J$ but) there is an oracle $\Pi^w_{j,i}$ which, with the same session ID with $\Pi^t_{i,j}$, has been revealed, $\mathcal{B}$ aborts the game (**Event 4**). Otherwise, $\mathcal{B}$ randomly chooses a bit string $\zeta \in \{0,1\}^\ell$ and gives it to $\mathcal{A}$ as the response.

Once $\mathcal{A}$ finishes the queries and returns its guess, $\mathcal{B}$ goes through $\mathbf{KDF2}^{list}$ and for each $T = g_{1t}, R = R_j$ from the tuple indexed by $\mathtt{ID}_i$ of the revealed oracle, if the revealed oracle is an initiator, otherwise $T = g_{2t}, R = R_i$ for the tuple indexed by $\mathtt{ID}_j$

– $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $([x]P_1, P_2, [h_0 + x]P_1, R, T)$.
– If $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ returns $T^{1/y}$ as the response to the Gap-q_1-BCAA1$_{1,2}$ challenge.
– If no match is found, $\mathcal{B}$ fails (**Event 5**, i.e., $\hat{e}([y]P_1, [\frac{x}{h_0+x}]P_2)$ has not been queried on **KDF2**).

Claim 1 *If algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.*

Proof: $\mathcal{B}$'s responses to $\mathbf{H2RF}_1$ queries are uniformly and independently distributed in $\mathbb{Z}^*_r$ as in the real attack. **KDF2** is modeled as a random oracle which requires that for each unique input, there should be only one response. We note that the simulation $\mathcal{B}$ substantially makes use of the programmability of random oracle and the access to the DBIDH oracle to guarantee that the response to the **KDF2** query is consistent with the Reveal query. The responses in other types of query are valid as well. Hence the claim follows.

Note the agreed key value in the chosen fresh oracle $\Pi^t_{i,j}$ should include $T = \hat{e}(R, D_i)$ where $R = [y]P_1$ and D_i is the private key of party j whose public key is $[h_0]P_1 + [x]P_1$, if the game does not abort.

Claim 2 $\Pr[\overline{\texttt{Event 5}}] \geq \epsilon(k)$.

The proof is similar to Claim 2 in [11]. We skip the details.

Let **Event 6** be that, in the attack, adversary B indeed chooses oracle $\Pi^J_{i,j}$ as the challenger oracle where $\mathtt{ID}_j$ has been queried on $\mathbf{H2RF}_1$ as the I-th distinct identifier query. Then following the rules of the game, it's clear that Event 1, 2, 3, 4 would not happen. So,

$$\Pr[\overline{(\texttt{Event 1} \vee \texttt{Event 2} \vee \texttt{Event 3} \vee \texttt{Event 4})}] = \Pr[\texttt{Event 6}] \geq \frac{1}{q_1 \cdot q_o}.$$

Overall, we have

$$\begin{aligned} \Pr[A \text{ wins}] &= \Pr[\texttt{Event 6} \wedge \overline{\texttt{Event 5}}] \\ &\geq \tfrac{1}{q_1 \cdot q_o}\epsilon(k). \end{aligned}$$

This completes the security proof. □

Theorem 2 *SM9-KA can be instantiated to achieve the master secret forward secrecy, provided that* **KDF2** *is a random oracle and the ψ-BDH$_{2,1}$ assumption is sound. Specifically, suppose that there is an adversary $\mathcal{A}$ with non-negligible probability $\epsilon(k)$ and running time $t(k)$ against the protocol that chooses generators P_1 and P_2 as the ψ-BDH$_{2,1}$ problem, and in the attack* **KDF2** *has been queried q_2 times, and q_o sessions including incomplete ones have been created. Then there exists an algorithm $\mathcal{B}$ solving the ψ-BDH$_{2,1}$ problem with advantage*

$$\mathtt{Adv}_{\mathcal{B}}(k) \geq \frac{\epsilon(k)}{q_2 \cdot q_o}$$

within a running time essentially same as $t(k)$.

Proof: Given an instance of the ψ-BDH$_{2,1}$ problem $(P_1, P_2, [a]P_1, [b]P_1)$ with a set of pairing parameter where there is no efficient group homomorphism ψ such that $\psi(P_1) = P_2$. $\mathcal{B}$ simulates $\mathbb{G}_{\mathtt{ID}}$ to generate the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, [x]P_1, \hat{e}([x]P_1, P_2), H_v, hid)$, i.e., using a randomly chosen $x \in \mathbb{Z}_r^*$ as the master secret key. Function **KDF2** is constructed from the hash function H_v and is simulated as a random oracle controlled by $\mathcal{B}$.

Without loss of generality, we use $\Pi_{i,j}^t$ to refer to the t-th session among all the sessions including incomplete ones created in the attack. $\mathcal{B}$ randomly chooses $1 \leq J \leq q_o$, and interacts with $\mathcal{A}$ in the following way:

- **KDF2**$(\mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t})$: $\mathcal{B}$ maintains a list $\mathbf{KDF2}^{list}$ of pairs in the form $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t} \rangle, \zeta_t)$. To respond to a query, $\mathcal{B}$ does the following operations:
  - If $\mathbf{KDF2}^{list}$ has a tuple indexed by $\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t} \rangle$, then $\mathcal{B}$ responds with ζ_t.
  - Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta_t \in \{0,1\}^\ell$ and inserts a new tuple $(\langle \mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_{1t}, g_{2t}, g_{3t} \rangle, \zeta_t)$ into the list $\mathbf{KDF2}^{list}$. It responds to $\mathcal{A}$ with ζ_t.
- Corrupt$(\mathtt{ID}_i)$: $\mathcal{B}$ returns $[\frac{x}{\mathbf{H2RF}_1(\mathtt{ID}_i)+x}]P_2$.
- Send$(\Pi_{i,j}^t, R)$: $\mathcal{B}$ maintains a list with tuples of $(\Pi_{i,j}^t, r_{i,j}^t, tran_{i,j}^t)$ and responds to the query as follows:
  - If $t \neq J$, $\mathcal{B}$ randomly chooses $r^t \in \mathbb{Z}_r^*$ as the random flips of the oracle and generates $[r^t]([\mathbf{H2RF}_1(\mathtt{ID}_j)]P_1 + [x]P_1)$ as the message.
  - Otherwise ($t = J$. Without loss of generality, let $\mathtt{ID}_I$ and $\mathtt{ID}_R$ be the identity of the initiator and responder of the session respectively),
    - If $R = \lambda$, $\mathcal{B}$ uses $[a]P_1$ as the message.
    - Otherwise, if $R \neq [a]P_1$, $\mathcal{B}$ aborts (**Event 1**), otherwise $\mathcal{B}$ uses $[b]P_1$ as the message.
- Reveal$(\Pi_{i,j}^t)$: $\mathcal{B}$ responds to the query as follows:
  - If $t = J$, then abort the game (**Event 2**).
  - Otherwise, compute $D_i = [\frac{x}{\mathbf{H2RF}_1(\mathtt{ID}_i)+x}]P_2$, $g_1 = \hat{e}(R_j, D_i)$, $g_2 = \hat{e}([x]P_1, P_2)^{r^t}$, $g_3 = g_1^{r^t}$ where R_j is the incoming message and r^t is the random flips of the oracle $\Pi_{i,j}^t$. $\mathcal{B}$ responds with **KDF2**$(\mathtt{ID}_i, \mathtt{ID}_j, R_i, R_j, g_1, g_2, g_3)$ if the oracle is the initiator, or **KDF2**$(\mathtt{ID}_j, \mathtt{ID}_i, R_j, R_i, g_2, g_1, g_3)$ otherwise.

– Test($\Pi_{i,j}^t$): If $t \neq J$, $\mathcal{B}$ aborts the game (**Event 3**). Otherwise, $\mathcal{B}$ randomly chooses a string $\zeta \in \{0,1\}^\ell$ and gives it to $\mathcal{A}$ as the response.

Once $\mathcal{A}$ finishes the queries and returns its guess, $\mathcal{B}$ goes through $\mathbf{KDF2}^{list}$ to find a set $\mathcal{L}$ of tuples indexed by $\langle \mathtt{ID}_I, \mathtt{ID}_R, [a]P_1, [b]P_1, \hat{e}([b]P_1, [\frac{x}{\mathbf{H2RF}_1(\mathtt{ID}_I)+x}]P_2), \hat{e}([a]P_1, [\frac{x}{\mathbf{H2RF}_1(\mathtt{ID}_R)+x}]P_2), * \rangle$. $*$ matches any values. $\mathcal{B}$ randomly chooses g_{3t} from $\mathcal{L}$ and returns $X = g_{3t}^{(\mathbf{H2RF}_1(\mathtt{ID}_I)+x)(\mathbf{H2RF}_1(\mathtt{ID}_R)+x)/x}$ as the answer to the ψ-BDH$_{2,1}$ problem.

Note the agreed key value in the chosen fresh oracle $\Pi_{i,j}^t$ should include $Y = g_{3t} = \hat{e}([b]P_1, [\frac{x}{\mathbf{H2RF}_1(\mathtt{ID}_I)+x}]P_2)^{r_I}$ where $[r_I]([\mathbf{H2RF}_1(\mathtt{ID}_R)]P_1 + [x]P_1) = [a]P_1$, if the game does not abort. Hence, if the value has been queried with **KDF2** and g_{3t} happens with probability at least $1/q_2$ to be the right choice from $\mathcal{L}$, X is the correct answer to the ψ-BDH$_{2,1}$ problem.

Claim 3 *Let* **Event 4** *be that Y along with identities and exchanged messages has not been queried with* **KDF2**. $\Pr[\overline{\texttt{Event 4}}] \geq \epsilon(k)$.

The proof is similar to Claim 2 in [11]. We skip the details.

Let **Event 5** be that, in the attack, adversary B indeed chose oracle $\Pi_{i,j}^J$ as the challenger oracle. Then, following the rules of the game by Definition 3, it's clear that Event 1, 2, 3 would not happen. So,

$$\Pr[\overline{(\texttt{Event 1} \vee \texttt{Event 2} \vee \texttt{Event 3})}] = \Pr[\texttt{Event 5}] \geq \frac{1}{q_o}.$$

Overall, we have

$$\begin{aligned} \Pr[A \text{ wins}] &= \Pr[\texttt{Event 5} \wedge \overline{\texttt{Event 4}}] \\ &\geq \tfrac{1}{q_2 \cdot q_o}\epsilon(k). \end{aligned}$$

This completes the security proof. □

7.3 Security Analysis of SM9-IBE

An identity-based encryption is specified by four algorithms [7]:

– **Setup** $\mathbb{G}_{\mathtt{ID}}(1^k)$: Given a security parameter k, the probabilistic algorithm outputs the master public key $M_{\mathfrak{pk}}$ and the master secret key $M_{\mathfrak{sk}}$.

$$(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^k)$$

– **Private-Key-Extract** $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$: The probabilistic algorithm takes as the input $M_{\mathfrak{pk}}, M_{\mathfrak{sk}}$ and the identifier string $\mathtt{ID}_A \in \{0,1\}^*$ for entity A, and outputs the private key D_A associated with $\mathtt{ID}_A$.

$$D_A \leftarrow \mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$$

– **Encrypt** $\mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, m)$: The probabilistic algorithm takes $M_{\mathfrak{pk}}$, $\mathtt{ID}_A$, the message m from the message space $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ as the inputs, and outputs a ciphertext C in the ciphertext space $\mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.

$$C \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, m)$$

– **Decrypt** $\mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A, C)$: The deterministic algorithm takes $M_{\mathfrak{pk}}$, $\mathtt{ID}_A$, D_A and C as input, and outputs the plaintext m or a failure symbol $\perp$ if C is invalid.

$$(m \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A, C)$$

Boneh and Franklin [7] formalized a security notion of IBE: ID-IND-CCA2 security, by the following two-stage game defined in Table 1 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the encryption algorithm and a challenger.

Table 1. IBE Security Formulation

ID-IND Adversarial Game
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^k)$.
2. $(st, \mathtt{ID}^*, m_0, m_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}})$.
3. $b \leftarrow \{0, 1\}$.
4. $C^* \leftarrow \mathbb{E}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}^*, m_b)$.
5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathfrak{pk}}, C^*, st, \mathtt{ID}^*, m_0, m_1)$.

In the games st is some state information and $\mathcal{O}_{\mathtt{ID}}$ denotes oracles to which the adversary has access. In the CCA2 attack model, the adversary has access to two oracles:

1. **Extraction**. A private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of $D_{\mathtt{ID}}$.
2. **Decryption**. A decryption oracle which, on input an identity $\mathtt{ID}$ and a ciphertext of the adversary's choice, will return the corresponding plaintext or $\perp$. This is subject to the restriction that in the second phase $\mathcal{A}_2$ is not allowed to call this oracle with the pair $(C^*, \mathtt{ID}^*)$.

The adversary's advantage in the game is defined to be

$$\mathtt{Adv}^{\mathtt{ID-IND-CCA2}}_{\mathtt{ID}-\mathcal{A}}(k) = \mid 2\Pr[b' = b] - 1 \mid .$$

Definition 4 *An IBE algorithm is considered to be ID-IND-CCA2 secure, if for all PPT adversaries, the advantage in the game is a negligible function of the security parameter k.*

Following up Cramer and Shoup's formalization of hybrid encryption [12], Bentahar el al. [3] extended the hybrid encryption to identity-based schemes. Their main result is that an ID-IND-CCA2 secure IBE can be constructed from an ID-IND-CCA2 secure ID-KEM and a secure DEM.

Similar to IBE, an ID-KEM scheme is specified by four algorithms as well.

- **Setup** $\mathbb{G}_{\mathtt{ID-KEM}}(1^k)$: The algorithm is the same as $\mathbb{G}_{\mathtt{ID}}(1^k)$.
- **Private-Key-Extract** $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID}_A)$: The algorithm is the same as $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}},\ M_{\mathfrak{sk}},\ \mathtt{ID}_A)$.
- **KEM-Encap** $\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_A)$: This probabilistic algorithm takes as input $M_{\mathfrak{pk}}$ and $\mathtt{ID}_A$, and outputs a key K in the key space $\mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$ and the encapsulation of the key C in the encapsulation space $\mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$.

$$(K, C) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_A)$$

- **KEM-Decap** $\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A, C)$: This deterministic algorithm takes as input $M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A$ and C, and outputs the encapsulated key K in C or a failure symbol $\perp$ if C is an invalid encapsulation.

$$(K \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_A, D_A, C),$$

Consider the two-stage game in Table 2 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the ID-KEM and a challenger.

Table 2. ID-KEM Security Formulation

ID-IND Adversarial Game
1. $(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID-KEM}}(1^k)$.
2. $(st, \mathtt{ID}^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathtt{ID-KEM}}}(M_{\mathfrak{pk}})$.
3. $(K_0, C^*) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}^*)$.
4. $K_1 \leftarrow \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$.
5. $b \leftarrow \{0, 1\}$.
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathtt{ID-KEM}}}(M_{\mathfrak{pk}}, C^*, st, \mathtt{ID}^*, K_b)$.

In the games st is some state information and $\mathcal{O}_{\mathtt{ID-KEM}}$ denotes oracles to which the adversary has access. In the CCA2 attack model, the adversary has access to two oracles::

1. **Extraction**. A private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of $D_{\mathtt{ID}}$.
2. **Decapsulation**. A decapsulation oracle which, on input an identity $\mathtt{ID}$ and encapsulation of the adversary's choice, will return the encapsulated key. This is subject to the restriction that in the second phase $\mathcal{A}_2$ is not allowed to call this oracle with the pair $(C^*, \mathtt{ID}^*)$.

The adversary's advantage is defined to be

$$\mathtt{Adv}^{\mathtt{ID-IND-CCA2}}_{\mathtt{ID-KEM}-\mathcal{A}}(k) = \mid 2\Pr[b' = b] - 1 \mid .$$

Definition 5 *An ID-KEM is considered to be ID-IND-CCA2 secure, if for all PPT adversaries $\mathcal{A}$, the advantage in the game above is a negligible function of the security parameter k.*

Apart from the above security requirement, it is required in this report that the ID-KEM has an extra property as follow. In an ID-KEM, for the pair (M_{pk}, M_{sk}) generated by the Setup algorithm and every $(\mathtt{ID}_A, D_A)$ where $\mathtt{ID}_A \in \{0,1\}^*$ and D_A is generated by the Private-Key-Extract algorithm using $(M_{pk}, M_{sk}, \mathtt{ID}_A)$, all encapsulations created with $(M_{pk}, \mathtt{ID}_A)$ decapsulate properly with (M_{pk}, D_A) (in other words, BadKeyPairs (Section 7.1 [12]) are negligibly few). It is easy to see that SM9-KEM presented in this report has this property.

In the hybrid encryption, a DEM uses the key generated by a KEM to encrypt the message. As the DEM uses a different key derived by the KEM to encrypt each message, a one-time symmetric-key encryption with proper security properties is sufficient for such purpose.

A one-time symmetric-key encryption consists of two deterministic polynomial-time algorithms with the key, message and ciphertext spaces defined by $\mathbb{K}_{\mathtt{SK}}(1^k)$, $\mathbb{M}_{\mathtt{SK}}(1^k)$ and $\mathbb{C}_{\mathtt{SK}}(1^k)$ given the security parameter k:

- **Encrypt** $\mathbb{E}_{\mathtt{SK}}(K, m)$: The encryption algorithm takes a secret key $K \in \mathbb{K}_{\mathtt{SK}}(1^k)$ and a message $m \in \mathbb{M}_{\mathtt{SK}}(1^k)$ as input, and outputs the ciphertext $C \in \mathbb{C}_{\mathtt{SK}}(1^k)$.
$$C \leftarrow \mathbb{E}_{\mathtt{SK}}(K, m)$$
- **Decrypt** $\mathbb{D}_{\mathtt{SK}}(K, C)$: Given a secret key K and a cipertext C, the algorithm outputs the plaintext m or a failure symbol $\perp$.
$$(m \text{ or } \perp) \leftarrow \mathbb{D}_{\mathtt{SK}}(K, C)$$

The two algorithms satisfy $\mathbb{D}_{\mathtt{SK}}(K, \mathbb{E}_{\mathtt{SK}}(K, m)) = m$ for $m \in \mathbb{M}_{\mathtt{SK}}(1^k)$ and $K \in \mathbb{K}_{\mathtt{SK}}(1^k)$.

The security of one-time symmetric-key encryption is defined by the Find-Guess (FG) game in Table 3 between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ of the DEM and a challenger:

In the game m_0 and m_1 are of equal length from the message space and *st* is some state information. $\mathcal{O}_{\mathtt{SK}}$ is the oracle that the adversary can access depending on the attack model. In the CCA attack model, the adversary has access to a decryption oracle.

- A decryption oracle which, on input a ciphertext C, returns $\mathbb{D}_{\mathtt{SK}}(K, C)$ with K chosen in Step 3 in the game.

The adversary's advantage in the game above is defined to be

$$\mathtt{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}-\mathcal{A}}(k) = \mid 2\Pr[b' = b] - 1 \mid .$$

Table 3. DEM Security Formulation

FG Adversarial Game
1. $(st, (m_0, m_1)) \leftarrow \mathcal{A}_1(1^k)$. 2. $b \leftarrow \{0,1\}$. 3. $K \leftarrow \mathbb{K}_{\mathtt{SK}}(1^k)$. 4. $C^* \leftarrow \mathbb{E}_{\mathtt{SK}}(K, m_b)$. 5. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}^{\mathtt{SK}}}(C^*, st, m_0, m_1)$.

Definition 6 *A one-time encryption is consider to be FG-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage in the game above is a negligible function of the security parameter k.*

The FG-CCA secure one-time encryptions are easy to get, such as the one-time pad encryption with a secure message authentication code algorithm [19, 12].

A hybrid IBE construction consisting of the sequential combination of an ID-KEM with a DEM proceeds as defined in Table 4. Here, it is assumed that the key space output by the KEM is identical with the secret key space used by the DEM.

Table 4. Hybrid IBE

$\mathbb{E}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}_A, m)$	$\mathbb{D}_{\mathtt{ID}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C)$
– $(K, C_1) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A)$ – $C_2 \leftarrow \mathbb{E}_{\mathtt{SK}}(K, m)$ – Return $C = \langle C_1, C_2 \rangle$	– Parse C as $\langle C_1, C_2 \rangle$ – $K \leftarrow \mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}}, \mathtt{ID}_A, D_A, C_1)$ – If $K = \perp$, return $\perp$ – $m \leftarrow \mathbb{D}_{\mathtt{SK}}(K, C_2)$ – Return m

Similar to the result of hybrid encryption in [12], Bentahar *et al.* obtained the following theorem concerning the security of hybrid IBE.

Theorem 3 *[Bentahar et al. [3]] Let $\mathcal{A}$ be a PPT ID-IND-CCA2 adversary of the IBE scheme $\mathcal{E}$ above. There exists PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, whose running time is essentially that of $\mathcal{A}$, such that*

$$\mathtt{Adv}^{\mathtt{ID-IND-CCA2}}_{\mathtt{ID}-\mathcal{A}}(k) \leq 2\mathtt{Adv}^{\mathtt{ID-IND-CCA2}}_{\mathtt{ID-KEM}-\mathcal{B}_1}(k) + \mathtt{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}-\mathcal{B}_2}(k).$$

Here, we only present the security analysis of SM9-KEM. The security of the full SM9-IBE follows from Theorem 3, 4 and the security result of DEM [19].

Theorem 4 *SM9-KEM is ID-IND-CCA2 secure provided that* $\mathbf{H2RF}_1, \mathbf{KDF2}$ *are random oracles and the Gap-*τ*-BCAA1*$_{1,2}$ *assumption is sound. Specifically, suppose there exists an ID-IND-CCA2 adversary* $\mathcal{A}$ *against SM9-KEM that has advantage* $\epsilon(k)$ *and running time* $t(k)$*, and suppose also that during the attack* $\mathcal{A}$ *makes at most* q_D *queries on the Decapsulation query,* $q_1 + 1$ *queries on* $\mathbf{H2RF}_1$ *and* q_2 *queries on* $\mathbf{KDF2}$ *with* $\mathtt{ID}_*$*. Then there exists an algorithm* $\mathcal{B}$ *solving the Gap-*q_1*-BCAA1*$_{1,2}$ *problem with advantage*

$$\mathtt{Adv}_{\mathcal{B}}(k) \geq \frac{\epsilon(k)}{q_1 + 1}$$

within running time

$$t_{\mathcal{B}}(k) \leq t(k) + O(q_2 \cdot q_D \cdot \mathcal{O}),$$

where $\mathcal{O}$ *is the time of one access to the DBIDH*$_{1,1}$ *oracle.*

Proof: Given an instance of the Gap-q_1-BCAA1$_{1,2}$ problem $(P_1, P_2, [x]P_1, h_0, (h_1, [\frac{x}{h_1+x}]P_2), \ldots, (h_{q_1}, [\frac{x}{h_{q_1}+x}]P_2))$ with a set of pairing parameter where $h_i \in_R \mathbb{Z}_r^*$ for $0 \leq i \leq q_1$ and the DBIDH$_{1,1}$ oracle $\mathcal{O}_{DBIDH}$, $\mathcal{B}$ simulates $\mathbb{G}_{\mathtt{ID-KEM}}$ to generate the system parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, P_1, P_2, [x]P_1, \hat{e}([x]P_1, P_2), H_v, hid)$, i.e., using x as the master secret key, which it does not know. Function $\mathbf{H2RF}_1$ and $\mathbf{KDF2}$ are constructed from the hash function H_v and are simulated as two random oracles controlled by $\mathcal{B}$.

$\mathcal{B}$ randomly chooses $1 \leq I \leq q_1 + 1$ and interacts with $\mathcal{A}$ as follows:

- $\mathbf{H2RF}_1(\mathtt{ID}_i)$: $\mathcal{B}$ maintains a list $\mathbf{H2RF}_1^{list}$ of tuples $(\mathtt{ID}_i, h_i, D_i)$ as explained below. When $\mathcal{A}$ queries the oracle $\mathbf{H2RF}_1$ on $\mathtt{ID}_i$, $\mathcal{B}$ responds as follows:
  - If $\mathtt{ID}_i$ is on $\mathbf{H2RF}_1^{list}$ in a tuple $(\mathtt{ID}_i, h_i, D_i)$, then $\mathcal{B}$ responds with $\mathbf{H2RF}_1(\mathtt{ID}_i) = h_i$.
  - Otherwise, if the query is on the I-th distinct ID, then $\mathcal{B}$ stores $(\mathtt{ID}_I, h_0, \perp)$ into the tuple list and responds with $\mathbf{H2RF}_1(\mathtt{ID}_I) = h_0$.
  - Otherwise, $\mathcal{B}$ selects a random integer h_i with $i > 0$ from the Gap-q_1-BCAA1$_{1,2}$ instance which has not been chosen before and stores $(\mathtt{ID}_i, h_i, [\frac{x}{h_i+x}]P_2)$ into the tuple list. $\mathcal{B}$ responds with $\mathbf{H2RF}_1(\mathtt{ID}_i) = h_i$.
- $\mathbf{KDF2}(C_i, X_i, \mathtt{ID}_i)$: $\mathcal{B}$ maintains a list $\mathbf{KDF2}^{list}$ of pairs in the form $(\langle C_i, X_i, \mathtt{ID}_i\rangle, K_i)$. To respond to a query on $(C_i, X_i, \mathtt{ID}_i)$, $\mathcal{B}$ does the following operations:
  - If a pair $(\langle C_i, X_i, \mathtt{ID}_i\rangle, K_i)$ is on the list, then $\mathcal{B}$ responds with K_i.
  - Otherwise, $\mathcal{B}$ looks through list $\mathbf{H2RF}_1^{list}$. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $\mathbf{H2RF}_1(\mathtt{ID}_i)$. Depending on the value of D_i for $\mathtt{ID}_i$ on $\mathbf{H2RF}_1^{list}$, $\mathcal{B}$ responds differently.
    - If $D_i = \perp$,
      - $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $([x]P_1, P_2, [h_0 + x]P_1, C_i, X_i)$.
      - If $\mathcal{O}_{DBIDH}$ returns 1 and a tuple index by $(C_i, \mathtt{ID}_i)$ appears on list $\mathcal{L}_D$ (a list maintained in the Decapsulation specified later), $\mathcal{B}$ returns K_i from the tuple after putting $(\langle C_i, X_i, \mathtt{ID}_i\rangle, K_i)$ into $\mathbf{KDF2}^{list}$.

· Otherwise, $\mathcal{B}$ randomly chooses a string $K_i \in \{0,1\}^\ell$ and inserts a new pair $(\langle C_i,\ X_i,\ \mathtt{ID}_i\rangle, K_i)$ into $\mathbf{KDF2}^{list}$, and if $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ also inserts $(C_i, \mathtt{ID}_i, K_i)$ into $\mathcal{L}_D$. It responds to $\mathcal{A}$ with K_i.

∗ Otherwise ($D_i \neq \perp$), $\mathcal{B}$ randomly chooses a string $K_i \in \{0,1\}^\ell$ and inserts a new pair $(\langle C_i,\ X_i,\ \mathtt{ID}_i\rangle, K_i)$ into the list. It responds to $\mathcal{A}$ with K_i.

– **Extraction**($\mathtt{ID}_i$)**:** $\mathcal{B}$ looks through list $\mathbf{H2RF}_1^{list}$. If $\mathtt{ID}_i$ is not on the list, $\mathcal{B}$ queries $\mathbf{H2RF}_1(\mathtt{ID}_i)$. $\mathcal{B}$ checks the value of D_i: if $D_i \neq \perp$, then $\mathcal{B}$ responds with D_i; otherwise, $\mathcal{B}$ aborts the game (**Event 1**).

– **Decapsulation**($\mathtt{ID}_i, C_i$)**:** $\mathcal{B}$ maintains list $\mathcal{L}_D$ of pairs in the form $(C_i, \mathtt{ID}_i, K_i)$. To respond to the query, $\mathcal{B}$ first looks through list $\mathbf{H2RF}_1^{list}$. If $\mathtt{ID}_i$ is not on the list, then $\mathcal{B}$ queries $\mathbf{H2RF}_1(\mathtt{ID}_i)$. Depending on the value of D_i for $\mathtt{ID}_i$ on $\mathbf{H2RF}_1^{list}$, $\mathcal{B}$ responds differently.

1. If $D_i \neq \perp$, then $\mathcal{B}$ first computes $g^r = \hat{e}(C_i, D_i)$, and then queries $K_i = \mathbf{KDF2}(C_i, g^r, \mathtt{ID}_i)$. $\mathcal{B}$ responds with K_i.
2. Otherwise ($D_i = \perp$), $\mathcal{B}$ takes following actions:
   (a) If a tuple indexed by $(C_i, \mathtt{ID}_i)$ is on $\mathcal{L}_D$, return K_i from the tuple.
   (b) Otherwise, $\mathcal{B}$ randomly chooses $K_i \in \{0,1\}^\ell$ and inserts $(C_i, \mathtt{ID}_i, K_i)$ into the list $\mathcal{L}_D$. Finally $\mathcal{B}$ returns K_i.

– **Challenge:** At some point $\mathcal{A}$'s first stage will terminate and it will return a challenge identity $\mathtt{ID}^*$. If $\mathcal{A}$ has not called $\mathbf{H2RF}_1$ with input $\mathtt{ID}^*$ then $\mathcal{B}$ does so for it. If the corresponding value of $D_{\mathtt{ID}^*}$ is not equal to $\perp$, then $\mathcal{B}$ aborts (**Event 2**). $\mathcal{B}$ chooses a random value of $y \in \mathbb{Z}_r^*$ and a random value $K^* \in \{0,1\}^\ell$, and returns $(K^*, [y]P_1)$ as the challenge. For simplicity, if $(\mathtt{ID}^*, [y]P_1)$ has been queried on the Decapsulation query, $\mathcal{B}$ tries another random r.

– **Guess:** Once $\mathcal{A}$ outputs its guess, $\mathcal{B}$ answers the Gap-q_1-BCAA1$_{1,2}$ challenge in the following way.

1. For each tuple $(\langle [y]P_1,\ X_j,\ \mathtt{ID}_*\rangle,\ K_j)$ in $\mathbf{KDF2}^{list}$, $\mathcal{B}$ queries $\mathcal{O}_{DBIDH}$ with $([x]P_1, P_2, [h_0 + x]P_1, [y]P_1, X_j)$. If $\mathcal{O}_{DBIDH}$ returns 1, $\mathcal{B}$ outputs $X_j^{1/y}$ as the answer to the Gap-q_1-BCAA1$_{1,2}$ problem.
2. If no tuple is found on the list, $\mathcal{B}$ fails (**Event 3**).

Claim 4 *If algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack.*

Proof: $\mathcal{B}$'s responses to $\mathbf{H2RF}_1$ queries are uniformly and independently distributed in $\mathbb{Z}_r^*$ as in the real attack because of the behavior of the Setup phase in the simulation. **KDF2** is modeled as a random oracle which requires that for each unique input, there should be only one response. We note that the simulation substantially makes use of the programmability of random oracle and the access to the DBIDH$_{1,1}$ oracle to guarantee that the response to the **KDF2** query is consistent with the Decapsulation query. There are two subcases in the simulation.

– The adversary queries on $\mathbf{KDF2}(C_i, X_i, \mathtt{ID}_i)$. If $(C_i, X_i, \mathtt{ID}_i)$ has not been queried before on **KDF2**, $\mathcal{B}$ should make sure that the response must be consistent with the possible existing response generated in the Decapsulation queries when $\mathtt{ID}_i = \mathtt{ID}_*$. $\mathcal{B}$ exploits the access to the $\mathrm{DBIDH}_{1,1}$ oracle by testing $\hat{e}(C_i, [\frac{x}{h_0+x}]P_2) \stackrel{?}{=} X_i$. If the equation holds, $\mathcal{B}$ returns the response to the Decapsulation query on (ID_i, C_i) if such query has been issued.
– The adversary queries the Decapsulation oracle on $(\mathtt{ID}^*, C_i)$. $\mathcal{B}$ cannot compute $X_i = \hat{e}(C_i, [\frac{x}{h_0+x}]P_2)$ (note that if the game does not abort, $D_{\mathtt{ID}^*} = [\frac{x}{h_0+x}]P_2$). If $\mathbf{KDF2}(C_i, X_i, \mathtt{ID}_i)$ has not been queried, i.e., (ID_i, C_i, K_i) is not on $\mathcal{L}_D$, $\mathcal{B}$ can respond with any random string K_i. Otherwise, $\mathcal{B}$ uses K_i from the tuple on $\mathcal{L}_D$ indexed by (ID_i, C_i) that is inserted by a **KDF2** query.

The responses in other types of query are valid as well. Hence the claim is founded.

We now evaluate the probability that $\mathcal{B}$ does not abort the game. `Event 3` implies that the value $\hat{e}(C^*, [\frac{x}{h_0+x}]P_2)$, which is the key value in the challenge encapsulation, is not queried on **KDF2** in the simulation. Since **KDF2** is a random oracle, $\Pr[\mathcal{A} \text{ wins}|\mathtt{Event\ 3}] = \frac{1}{2}$. We have

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins}|\mathtt{Event\ 3}]\Pr[\mathtt{Event\ 3}] + \Pr[\mathcal{A} \text{ wins}|\overline{\mathtt{Event\ 3}}]\Pr[\overline{\mathtt{Event\ 3}}] \\ &\leq \frac{1}{2}(1 - \Pr[\overline{\mathtt{Event\ 3}}]) + \Pr[\overline{\mathtt{Event\ 3}}] = \frac{1}{2} + \frac{1}{2}\Pr[\overline{\mathtt{Event\ 3}}]. \\ \Pr[\mathcal{A} \text{ wins}] &\geq \Pr[\mathcal{A} \text{ wins}|\mathtt{Event\ 3}]\Pr[\mathtt{Event\ 3}] \\ &= \frac{1}{2}(1 - \Pr[\overline{\mathtt{Event\ 3}}]) = \frac{1}{2} - \frac{1}{2}\Pr[\overline{\mathtt{Event\ 3}}].\end{aligned}$$

So, we have $\Pr[\overline{\mathtt{Event\ 3}}] \geq \epsilon(k)$. Note that $\overline{\mathtt{Event\ 2}}$ implies $\overline{\mathtt{Event\ 1}}$ because of the rules of the game. Overall, we have

$$\Pr[\mathcal{B} \text{ wins}] = \Pr[\overline{\mathtt{Event\ 3}} \wedge \overline{\mathtt{Event\ 2}}] \geq \frac{\epsilon(k)}{q_1 + 1}.$$

This completes the security analysis of SM9-KEM. □

8 Performance Evaluation

Here we briefly compare the performance of SM9 with the identity-based signature schemes included in ISO/IEC 14888-3 [17], identity-based key agreements included in ISO/IEC 11770-3 [16] and encryption schemes in ISO/IEC 18033-5 [20]. Table 5 shows that SM9-IBS is more efficient than those two IBS schemes in ISO/IEC 14888-3. Table 6 shows that SM9-KA is more efficient than those two IB-KA protocols in ISO/IEC 11770-3 [16]. Table 7 shows that SM9-KEM maintains better performance in terms of both the computation efficiency and the cipher text size than those three schemes in ISO/IEC 18033-5.

Table 5. Performance of IBS Schemes from Pairings

	IBS1 [17]	IBS2 [17]	SM9-IBS
Private Key Extract			
Hash to $\mathbb{G}_1$	1	1	
Mul in $\mathbb{G}_1$	1	1	$\overline{1}$
Sign			
Mul in $\mathbb{G}_1$	$\overline{1}$	$\overline{2}^{(2)}$	$\overline{1}$
Exp in $\mathbb{G}_T$	$\overline{1}^{(1)}$		$\overline{1}$
Verify			
Hash to $\mathbb{G}_1$	1	1	
Mul in $\mathbb{G}_1$	$1^{(3)}$	1	
Mul in $\mathbb{G}_2$			$\overline{1}$
Exp in $\mathbb{G}_T$			$\overline{1}$
Pairings	2	2	1
Signature Size	$\lambda + \gamma_1$	$2\gamma_1$	$\lambda + \gamma_1$

1. Assume exponentiation in $\mathbb{G}_T$ is faster than pairing and pre-computation of pairing with the signer's private key and the master public key is available.
2. Assume Y is pre-computed in producing the pre-signature [17] which is reasonable for a signer.
3. Assume multiplication in $\mathbb{G}_1$ is faster than exponentiation in $\mathbb{G}_T$.
4. Symbols $\overline{m}$ and n denote m fix-based multiplications or exponentiations and n general operations respectively.
5. Symbols λ, γ_1 denote the length of an element in $\mathbb{Z}_r^*$ and $\mathbb{G}_1$ respectively.

Table 6. Performance of IB-KA Protocols from Pairings

	SCC [16]	FSU [16]	SM9-KA
Private Key Extract			
Hash to $\mathbb{G}_2$	1	$1^{(1)}$	
Mul in $\mathbb{G}_2$	1	1	$\overline{1}$
Message Exchange			
Mul in $\mathbb{G}_1$	$\overline{1}$	$\overline{1}$	$\overline{2}$
Session Key Generation			
Hash to $\mathbb{G}_2$	1	1	
Mul in $\mathbb{G}_1$	$1+\overline{1}$	$1+\overline{1}$	
Exp in $\mathbb{G}_T$			$1+\overline{1}$
Pairings	2	2	1
Message Size	γ_1	γ_1	γ_1

1. The FSU scheme requires $\mathbb{G}_1 = \mathbb{G}_2$.

Table 7. Performance of IBE Schemes from Pairings

	BF-IBE [20]	BB_1-KEM [20]	SK-KEM [20]	SM9-KEM
Private Key Extract				
Hash to $\mathbb{G}_1$	$1^{(1)}$			
Mul in $\mathbb{G}_1$	1			
Mul in $\mathbb{G}_2$		$\overline{2}$	$\overline{1}$	$\overline{1}$
Encapsulate				
Hash to $\mathbb{G}_1$	1			
Mul in $\mathbb{G}_1$	1	$\overline{3}$	$\overline{2}$	$\overline{2}$
Mul in $\mathbb{G}_2$	$\overline{1}$			
Exp in $\mathbb{G}_T$		$\overline{1}$	$\overline{1}$	$\overline{1}$
Pairings	1			
Decapsulate				
Mul in $\mathbb{G}_1$			$\overline{1}^{(2)}$	
Mul in $\mathbb{G}_2$	$\overline{1}$			
Pairings	1	2	1	1
Full Cipher Text Size	$\gamma_2 + 2\zeta$	$2\gamma_1 + \eta$	$\gamma_1 + \delta + \eta$	$\gamma_1 + \eta$

1. It appears that when asymmetric pairings are used, mapping an identity string to an element in $\mathbb{G}_2$ instead of $\mathbb{G}_1$ could make BF-IBE more efficient. Here we strictly follow the specification in [20] for comparison.
2. Assume Q is pre-computed in KEM-Decrypt [20] which is reasonable for a decryptor.
3. Symbols $\gamma_i, \delta, \zeta, \eta$ denote the length of an element in $\mathbb{G}_i$, a random string, a plain text and a DEM respectively.

References

1. P.S.L.M. Barreto, B. Libert, N. McCullagh, and J. Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Proc. of Advances in cryptology ASIACRYPT05*, LNCS 3778, pp.515-532, 2005.
2. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proc. of Advances in Cryptology – Crypto '93*, LNCS 773, pp. 232–249, 1993.
3. K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. *J. of Cryptology*, Vol 21, pp. 178–199, 2008.
4. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proc. of Cryptography and Coding*, LNCS 1355, pp. 30–45, 1997.
5. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 223–238, 2004.
6. D. Boneh and X. Boyen. Short signatures without random oracles. In *Proc. of Advances in Cryptology - Eurocrypt 2004*, LNCS 3027, pp. 56–73, 2004.
7. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Proc. of Advances in Cryptology - Crypto 2001*, LNCS 2139, pp. 213–229, 2001.
8. S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings - The role of ψ revisited. *Discrete Applied Mathematics*, Vol 159, pp. 1311–1322, 2011.
9. L. Chen and Z. Cheng. Security proof of Sakai-Kasahar's identity-based encryption scheme. In *Proc. of Cryptography and Coding 2005*, LNCS 3796, pp. 442–459, 2005.
10. L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, Vol 6, pp. 213–241, 2007.
11. Z. Cheng and L. Chen. On security proof of McCullagh-Barreto's key agreement protocol and its variants. *International Journal of Security and Networks - Special Issue on Cryptography in Networks*, Vol 2, pp. 251–259, 2007.
12. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, Vol 33, pp. 167–226, 2003.
13. S. Galbraith, F. Hess, and F. Vercauteren. Aspects of pairing inversion. *IEEE Transactions on Information Theory*, Vol 54, Issue 12, pp. 5719–5728, 2008.
14. S. Galbraith, K. Paterson, and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, Vol 156, pp. 3113–3121, 2008.
15. GM/T 0044-2016. Identity-based cryptographic algorithms SM9. 2016.
16. ISO/IEC. Information technology – Secruity techniques – Key management – Part 3: Mechanisms using asymmetric techniques. *ISO/IEC 11770-3:2015*.
17. ISO/IEC. Information technology – Secruity techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms. *ISO/IEC 14888-3:2015*.
18. ISO/IEC. Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 5: Elliptic curve generation. *ISO/IEC 15946-5:2009*.
19. ISO/IEC. Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. *ISO/IEC 18033-2:2006*.
20. ISO/IEC. Information technology – Security techniques – Encryption algorithms – Part 5: Identity-based ciphers. *ISO/IEC 18033-5:2015*.

21. E. Lee, H. Lee, and C. Park. Efficient and generalized pairing computation on abelian varieties. In *IEEE Transactions on Information Theory*, Vol 55, pp. 1793–1803, 2009.
22. N. McCullagh and P.S.L.M. Barreto. A new two-party identity-based authenticated key agreement. In *Proc. of Topics in Cryptology CT-RSA 2005*, LNCS 3376, pp. 262-274, 2005.
23. N. McCullagh and P.S.L.M. Barreto. A new two-party identity-based authenticated key agreement. IACR ePrint Report 2004/122. Feb. 2005.
24. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2), pp. 481–484, 2002.
25. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. of Advances in Cryptology - Eurocrypt '97*, LNCS 1233, pp. 256–266, 1997.
26. F. Vercauteren. Optimal pairings. In *IEEE Transactions on Information Theory*, Vol 56, Issue 11, pp. 455–461, 2010.