Identity-Based Authenticated Asymmetric Group Key Agreement Protocol

Lei Zhang[1], Qianhong Wu[1,2], Bo Qin[1,3], Josep Domingo-Ferrer[1]

[1] Universitat Rovira i Virgili, Dept. of Comp. Eng. and Maths
UNESCO Chair in Data Privacy, Tarragona, Catalonia
[2] Wuhan University, School of Computer
Key Lab. of Aerospace Information Security and Trusted Computing
Ministry of Education, China
[3] Xi'an University of Technology, School of Science, Dept. of Maths, China
Email: {lei.zhang,qianhong.wu,bo.qin,josep.domingo}@urv.cat

Abstract. In identity-based public-key cryptography, an entity's public key can be easily derived from its identity. The direct derivation of public keys in identity-based public-key cryptography eliminates the need for certificates and solves certain public key management problems in traditional public-key cryptosystems. Recently, the notion of *asymmetric group key agreement* was introduced, in which the group members merely negotiate a common encryption key which is accessible to any entity, but they hold respective secret decryption keys. In this paper, we first propose a security model for identity-based authenticated asymmetric group key agreement (IB-AAGKA) protocols. We then propose an IB-AAGKA protocol which is proven secure under the Bilinear Diffie-Hellman Exponent assumption. Our protocol is also efficient, and readily adaptable to provide broadcast encryption.

Keywords: Identity-Based Public-Key Cryptography, Group Key Agreement, Asymmetric Group Key Agreement, Bilinear Map.

1 Introduction

Group Key Agreement (GKA) protocols are widely employed in many modern collaborative and distributed applications such as multi-party computations, audio/video conference and chat systems. Their main goal is to implement secure broadcast channels. In the conventional GKA definition, a group of members interact over an open network to establish a common secret key to be used to achieve secure broadcast. This secret key is shared by all group members. A limitation of conventional GKA systems is that only group members are allowed to broadcast to other group members. However, in practice, anyone is likely to be a potential sender, just as anyone can encrypt a message in public-key encryption. Observing this fact, recently, Wu *et al.* [25] introduced the notion of Asymmetric Group Key Agreement (AGKA). By their definition, instead of a common secret key, the group members merely negotiate a common encryption key which is accessible to any entity, but they hold respective secret decryption keys.

1.1 Motivation and Contribution of This Paper

In the real world, sometimes the bandwidth is not critical for GKA protocols but the round efficiency is. One-round key agreement protocols have several advantages [19,25] over key agreement protocols in two or more rounds. For instance, imagine a group of friends who wish to share their personal documents via the open network. If a two-round key agreement protocol is

employed to establish a secure channel, all friends should be online at the same time. However, if this group of friends live in different time zones, it is difficult for them to be online concurrently.

A trivial way to achieve one-round AGKA is for each member in the group to publish a public key and reserve the respective secret key. To send a message to this group, a sender separately encrypts for each member and generates the final ciphertext by concatenating all the underlying individual ones. It is easy to see that this trivial solution leads to a long ciphertext and forces the sender to store all the public keys of the group members. Instead of this trivial solution, Wu *et al.* [25] proposed a one-round AGKA protocol from *scratch*, which means that the protocol participants do not hold any secret values prior to the execution of the protocol.

Though the protocols from scratch are efficient, they are only secure against passive adversaries who just eavesdrop the communication channel. Active adversaries are more powerful since they are assumed to have a complete control over the communication channel. They have the ability to relay, delay, modify, interleave or delete the message flows during the execution of the protocol. In particular, an active adversary is able to mount well-known man-in-the-middle attacks. Hence, it is vital for an AGKA protocol to withstand the attacks from active adversaries. This calls for authenticated key agreement protocols.

An authenticated key agreement protocol is a key agreement protocol which aims to ensure that no entities aside from the intended ones can possibly compute the session key agreed. Authenticated key agreement protocols may be designed under different public-key cryptosystems. A number of key agreement protocols have been proposed under the traditional PKI-based public-key paradigm. In that paradigm, key agreement protocols rely on the entities obtaining each other's certificates, extracting each other's public keys, checking certificate chains (which may involve many signature verifications) and finally generating a shared session key. Furthermore, the management of public-key certificates requires a large amount of computation, storage, and communication. To eliminate such costs, Identity-Based Public Key Cryptography (IB-PKC) was introduced by Shamir [24] in 1984. The main feature of IB-PKC is that the public key of an entity can be easily derived from its identity, such as its telephone number or email address; the corresponding private key can only be derived by a trusted Private Key Generator (PKG) who owns the *master secret* of the system.

In this paper, we first specify a security model that an Identity-Based Authenticated Asymmetric Group Key Agreement (IB-AAGKA) protocol should satisfy. Our model allows an adversary to adaptively choose his targets, and it captures the IB version of the (modified) common security requirements (*e.g.*, *secrecy*, *known-key security* and *forward secrecy*), which are usually considered in GKA protocols. These newly defined security requirements are described as follows:

- Secrecy requires that only the legitimate participants (group members) can read the messages encrypted by the negotiated public key.
- Known-key security means that, if an adversary learns the group encryption/decryption keys of other sessions, he cannot compute subsequent group decryption keys.
- Forward secrecy ensures that the disclosure of long-term private keys of group members must not compromise the secrecy of the decryption keys established in earlier protocol runs. Specifically, we say a key agreement protocol offers *perfect forward secrecy* if the long-term private keys of all the group members involved may be compromised without compromising any group decryption key previously established by these group members. We say a key agreement offers *partial forward secrecy* if compromise of the long-term keys of one or more

specific group members does not compromise the group decryption keys established by these group members.

We also propose a non-trivial one round IB-AAGKA protocol satisfying our security requirements, which we prove in the random oracle model [3].

Our protocol is based on a specific identity-based multi-signature scheme which we call identity-based batch multi-signature (IB-B-MS), which may itself be interesting in its own right. Our scheme allows x signers to sign t messages in such a way that the length of the batch multi-signature consists only of $t + 1$ group elements. Furthermore, the batch multi-signature can be separated into t individual multi-signatures.

1.2 Related Work

Since Diffie and Hellman published their solution to key agreement [14], much attention has been paid to this primitive. Joux [19] was the first who extended key agreement to three parties. We notice that both the Diffie-Hellman and Joux protocols are one-round key agreement protocols. However, when the protocol participants are more than three, it seems knotty to construct key agreement protocols without additional rounds. Over the years, many attempts have been made at extending the Diffie-Hellman and Joux protocols to n parties. Among them, the Burmester-Desmedt protocol [11] is one of the best-known. This protocol requires two rounds and is the most efficient existing GKA protocol in round efficiency without constraints on n.

For a key agreement protocol to be usable in open networks, it should be resistant against active adversaries. However, the basic Diffie-Hellman and Joux protocols as well as the Burmester-Desmedt protocol do not authenticate the communication entities. Hence they are not suited for hostile networks where man-in-the-middle attacks may happen. Several protocols have been proposed to add authentication [13,22,23]; among those, the GKA protocol in [13] is based on IB-PKC. This protocol refers to Katz and Yung's results [20] for an authenticated version and requires two rounds.

The paradigm of provable security subsumes an abstract formalization that considers the protocol environment and identifies its security goals. Bresson *et al.* [10] were the first to formalize the security model for group key agreement protocols. Their model is based on the previous security model for key agreement protocols between two or three parties [1,2,4]. Later, this model was refined by Bresson *et al.* [8,9] and some variants [20,21] of it appeared. These models are widely accepted in proving the security of GKA protocols. In this paper, we will borrow some ideas from these models to define the security model for IB-AAGKA protocols.

1.3 Paper Outline

We organize the rest of the paper as follows. Section 2 reviews bilinear maps and some complexity assumptions. Section 3 defines the security of IB-AAGKA protocols. Our identity-based batch multi-signature is introduced in Section 4. Section 5 describes our IB-AAGKA protocol. Finally, we conclude in Section 6.

2 Bilinear Maps and Complexity Assumptions

We review bilinear maps and related complexity assumptions in this section. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of prime order q, and g be a generator of $\mathbb{G}_1$. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is called a bilinear map if it satisfies the following properties:

1. Bilinearity: $\hat{e}(g^\beta, g^\gamma) = \hat{e}(g,g)^{\beta\gamma}$ for all $\beta, \gamma \in \mathbb{Z}_q^*$.
2. Non-degeneracy: There exists $u, v \in \mathbb{G}_1$ such that $\hat{e}(u,v) \neq 1$.
3. Computability: There exists an efficient algorithm to compute $\hat{e}(u,v)$ for any $u, v \in \mathbb{G}_1$.

The security of our protocol is based on the hardness of the computational Diffie-Hellman (CDH) problem and the k-Bilinear Diffie-Hellman Exponent (BDHE) problem [5], which are as follows:

CDH Problem: Given g, g^α, g^β for unknown $\alpha, \beta \in \mathbb{Z}_q$, compute $g^{\alpha\beta}$.

CDH Assumption: Let $\mathcal{B}$ be an algorithm which has advantage

$$\mathsf{Adv}(\mathcal{B}) = \Pr\left[\mathcal{B}(g, g^\alpha, g^\beta) = g^{\alpha\beta}\right]$$

in solving the CDH problem. The CDH assumption is that $\mathsf{Adv}(\mathcal{B})$ is negligible for any polynomial-time algorithm $\mathcal{B}$.

k-BDHE Problem: Given g, h, and $y_i = g^{\alpha^i}$ in $\mathbb{G}_1$ for $i = 1, 2, ..., k, k+2, ..., 2k$ as input, compute $\hat{e}(g,h)^{\alpha^k}$. Since the input vector is missing the term $g^{\alpha^{k+1}}$, the bilinear map does not seem to help computing $e(g,h)^{\alpha^{k+1}}$.

k-BDHE Assumption: Let $\mathcal{B}$ be an algorithm which has advantage

$$\mathsf{Adv}(\mathcal{B}) = \Pr\left[\mathcal{B}(g, h, y_1, ..., y_k, y_{k+2}, ..., y_{2k}) = e(g,h)^{\alpha^{k+1}}\right]$$

in solving k-BDHE problem. The k-BDHE assumption is that $\mathsf{Adv}(\mathcal{B})$ is negligible for any polynomial-time algorithm $\mathcal{B}$.

3 Security Model

The first security model for AGKA protocols was presented by Wu *et al.* [25], derived from the security model for conventional GKA protocols [10]. We note that the security model in [25] only considers passive attackers. In the sequel, we will extend this model to capture the ability of active attackers and integrate the notion of IB-PKC.

3.1 Participants and Notations

Let $\mathbb{P}$ be a set with polynomial-size of potential protocol participants. Each participant in $\mathbb{P}$ has an identity and a private key. Any subset $\mathbb{U} = \{\mathcal{U}_1, ..., \mathcal{U}_n\} \subseteq \mathbb{P}$ may decide at any point to establish a confidential channel among them. We use $\Pi^\pi_{\mathcal{U}_i}$ to represent instance π of participant $\mathcal{U}_i$ involved with partner participants $\{\mathcal{U}_1, ..., \mathcal{U}_{i-1}, \mathcal{U}_i + 1, \mathcal{U}_n\}$ in a session. Each instance $\Pi^\pi_{\mathcal{U}_i}$ holds the variables $\mathsf{pid}^\pi_{\mathcal{U}_i}$, $\mathsf{sid}^\pi_{\mathcal{U}_i}$, $\mathsf{ms}^\pi_{\mathcal{U}_i}$, $\mathsf{ek}^\pi_{\mathcal{U}_i}$, $\mathsf{dk}^\pi_{\mathcal{U}_i}$ and $\mathsf{state}^\pi_{\mathcal{U}_i}$ which are defined below:

- $\mathsf{pid}^\pi_{\mathcal{U}_i}$ is the *partner ID* of instance $\Pi^\pi_{\mathcal{U}_i}$. It is a set containing the identities of the participants in the group with whom $\Pi^\pi_{\mathcal{U}_i}$ intends to establish a session key including $\mathcal{U}_i$ itself. For simplicity, we assume that the identities in $\mathsf{pid}^\pi_{\mathcal{U}_i}$ are lexicographically ordered.
- $\mathsf{sid}^\pi_{\mathcal{U}_i}$ is the *session ID* of instance $\Pi^\pi_{\mathcal{U}_i}$. We follow [21] in assuming that unique session IDs are provided by some higher-level protocol when the group key-exchange protocol is first initiated. Therefore, all members taking part in a given execution of a protocol will have the same session ID.

- $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$ is the concatenation of all messages sent and received by $\Pi^{\pi}_{\mathcal{U}_i}$ during its execution, where the messages are ordered by round, and within each round lexicographically by the identities of the purported senders.
- $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ is the encryption key held by $\Pi^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{dk}^{\pi}_{\mathcal{U}_i}$ is the decryption key held by $\Pi^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{state}^{\pi}_{\mathcal{U}_i}$ represents the current (internal) state of instance $\Pi^{\pi}_{\mathcal{U}_i}$. When an instance has *terminated*, it is done sending and receiving messages. We say that an IB-AAGKA protocol has been *successfully terminated* (accepted) in the instance $\Pi^{\pi}_{\mathcal{U}_i}$ if it possesses $\mathsf{ek}^{\pi}_{\mathcal{U}_i}(\neq null)$, $\mathsf{dk}^{\pi}_{\mathcal{U}_i}(\neq null)$, $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$.

Definition 1 (Partnering). *We say instances $\Pi^{\pi}_{\mathcal{U}_i}$ and $\Pi^{\pi'}_{\mathcal{U}_j}$ (with $i \neq j$) are partnered iff (1) they are successfully terminated; (2) $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \mathsf{pid}^{\pi'}_{\mathcal{U}_j}$; and (3) $\mathsf{sid}^{\pi}_{\mathcal{U}_i} = \mathsf{sid}^{\pi'}_{\mathcal{U}_j}$.*

3.2 The Model

In GKA protocols, secrecy is the core security definition. In conventional GKA protocols, secrecy is defined by the indistinguishability of the shared common secret key from a random string in the secret key space. However, in our IB-AAGKA, what is negotiated is only a common public encryption key while the group members' secret decryption keys are different. Observe that both conventional GKAs and our IB-AAGKA have the similar final goal of establishing a confidential broadcast channel among users. Hence, we directly use the confidentiality of the final broadcast channel to define the secrecy of an IB-AAGKA protocol. That is, secrecy is defined by the indistinguishability of a message encrypted under the negotiated public key from a random string in the ciphertext space. Specifically, we use the following game which is run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ who has full control of the network communications to define the security of IB-AAGKA protocols. This game has three stages which are described in detail as follows:

Initial: At this stage, the challenger $\mathcal{C}$ first runs $\mathsf{Setup}(\ell)$ to generate the system parameters params and $\mathsf{master\text{-}secret}$, then gives the resulting params to the adversary $\mathcal{A}$ while keeping $\mathsf{master\text{-}secret}$ secret.

Training: $\mathcal{C}$ is probed by $\mathcal{A}$ who can make the following queries:

- $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Delta)$[4]: Send message Δ to instance $\Pi^{\pi}_{\mathcal{U}_i}$, and output the reply generated by this instance. If $\Delta = (\mathsf{sid}, \mathsf{pid})$, this query prompts $\mathcal{U}_i$ to initiate the protocol using session ID sid and partner ID pid. Note that the identity of $\mathcal{U}_i$ should be in pid, and if Δ is incorrect the query returns *null*.
- $\mathsf{Corrupt}(\mathcal{U}_i)$: Output the private key of participant $\mathcal{U}_i$. We will use it to model (partial) forward secrecy.
- $\mathsf{Ek.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: Output the encryption key $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: Output the decryption key $\mathsf{dk}^{\pi}_{\mathcal{U}_i}$. We will use it to model *known-key security*.

[4] Some models allow the adversary to make $\mathsf{Execute}$ queries. This feature is used to model passive attacks, where the adversary eavesdrops on honest execution of a group key agreement protocol. One may note that, if a GKA protocol is secure against active adversaries, the protocol is also secure against passive adversaries. Furthermore, as mentioned in [20], the $\mathsf{Execute}$ query can be simulated via repeated calls to the Send queries. Hence, in this paper, we do not consider $\mathsf{Execute}$ queries.

– $\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: At some point, $\mathcal{A}$ returns two messages (m_0, m_1) ($|m_0| = |m_1|$) and an instance $\Pi^{\pi}_{\mathcal{U}_i}$. It is required that $\Pi^{\pi}_{\mathcal{U}_i}$ be fresh (see Definition 2). $\mathcal{C}$ chooses a bit $b \in \{0, 1\}$ uniformly at random, encrypts m_b under $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ to produce the ciphertext c, and returns c to $\mathcal{A}$. Notice that $\mathcal{A}$ can submit this query only once, and we will use this query to model *Secrecy*.

Response: $\mathcal{A}$ returns a bit b'. We say that $\mathcal{A}$ wins if $b' = b$. $\mathcal{A}$'s advantage is defined to be $\mathsf{Adv}(\mathcal{A}) = |2\Pr[b = b'] - 1|$.

Definition 2 (Freshness). *An instance $\Pi^{\pi}_{\mathcal{U}_i}$ is fresh if none of the following happens:*

1. *At some point, $\mathcal{A}$ queried $\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$ or $\mathsf{Dk.Reveal}(\Pi^{\pi'}_{\mathcal{U}_j})$, where $\Pi^{\pi'}_{\mathcal{U}_j}$ is partnered with $\Pi^{\pi}_{\mathcal{U}_i}$.*
2. *A query $\mathsf{Corrupt}(\mathcal{U}_i)$ was asked before a query of the form $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Delta)$.*
3. *All the private keys of the participants with $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$ are corrupted. Since we do not allow $\mathcal{A}$ to corrupt all the participants in the same session, our game captures partial forward secrecy.*

Definition 3. *An IB-AAGKA protocol is said to be secure against semantically indistinguishable chosen identity and plaintext attacks (Ind-ID-CPA), if no randomized polynomial-time adversary has a non-negligible advantage in the above game. In other words, any randomized polynomial-time Ind-ID-CPA adversary $\mathcal{A}$ has an advantage*

$$\mathsf{Adv}(\mathcal{A}) = |2\Pr[b = b'] - 1|$$

that is negligible.

In this paper, we only consider security against chosen-plaintext attacks (CPA) for our IB-ASGKA protocol. To achieve security against chosen-ciphertext attacks (CCA), there are some generic approaches that convert a CPA secure encryption scheme into a CCA secure one, such as the Fujisaki-Okamoto conversion [16,6].

4 Building Block

In this section, we propose the signature scheme which will be used in our IB-AAGKA protocol. Our signature scheme can be viewed as a special identity-based multi-signature scheme which we call identity-based batch multi-signature (IB-B-MS) scheme. In our scheme, each signer will use a single random value to generate t signatures on t different messages. This way, the resulting signature (referred to as batch signature) on t messages of a signer only consists of $t + 1$ group elements. Furthermore, our scheme allows signatures on the same message from x signers to be aggregated into an IB-B-MS of $t + 1$ group elements. We notice that, when $t = 1$, our scheme degenerates into the multi-signature of Gentry and Ramzan [17].

4.1 Definition

An IB-B-MS scheme consists of the following five algorithms:

– BM.Setup: This algorithm takes as input a security parameter ℓ to generate a master-secret and a list of system parameters.
– BM.Extract: This algorithm takes as input an entity's identity ID_i, and the master-secret to produce the entity's private key.

- Sign: On input t messages, a signer's identity ID_i and private key s_i, this algorithm outputs a batch signature.
- Aggregate: On input a collection of x batch signatures on t messages from x signers, this algorithm outputs a batch multi-signature.
- BM.Verify: This algorithm is used to check the validity of a batch multi-signature. It outputs "all valid" if the batch multi-signature is valid; otherwise, it outputs an index set, which means that the multi-signatures on the messages with indices in that set are invalid.

4.2 The Model

The security of an IB-B-MS scheme is modeled via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Initial: $\mathcal{C}$ first runs BM.Setup to obtain a master-secret and the system parameter list params, then sends params to the adversary $\mathcal{A}$ while keeping the master-secret secret.

Training: The adversary $\mathcal{A}$ can perform a polynomially bounded number of the following types of queries in an adaptive manner.

- Extract: $\mathcal{A}$ can request the private key of an entity with identity ID_i. In response, $\mathcal{C}$ outputs the private key of this entity.
- Sign: $\mathcal{A}$ can request an entity's batch signature on n messages. On receiving such a query, $\mathcal{C}$ outputs a batch signature on those messages.

Forgery: $\mathcal{A}$ outputs a set of x entities whose identities form the set $\mathbb{L}^*_{ID} = \{ID^*_1, ..., ID^*_x\}$, a message m^* and a multi-signature σ^*. We say that $\mathcal{A}$ wins the above game if the following conditions are satisfied:

1. σ^* is a valid multi-signature on message m^* under identities $\{ID^*_1, ..., ID^*_x\}$.
2. At least one of the identities in $\mathbb{L}^*_{ID}$ has never been submitted during the BM.Extract queries and m^* together with that identity is not involved in the Sign queries.

Definition 4. *An IB-B-MS scheme is existentially unforgeable under adaptively chosen-message attack if and only if the success probability of any polynomially bounded adversary in the above game is negligible.*

4.3 The Scheme

The construction comes as follows.

- BM.Setup: On input a security parameter ℓ, the KGC chooses two cyclic multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime order q, where $\mathbb{G}_1$ is generated by g and there exists a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$. The KGC also chooses a random $\kappa \in \mathbb{Z}^*_q$ as the master-secret and sets $g_1 = g^\kappa$, and chooses cryptographic hash functions $H_1, H_2 : \{0,1\}^* \longrightarrow \mathbb{G}_1$. The system parameter list is params $= (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, H_1, H_2)$.
- BM.Extract: This algorithm takes as input master-secret κ and an entity's identity $ID_i \in \{0,1\}^*$. It generates the private key for the entity as follows:
  1. Compute $id_i = H_1(ID_i)$.
  2. Output the private key $s_i = id_i^\kappa$.

– Sign: To sign t messages $m_1, ..., m_t$, a signer with identity ID_i and private key s_i performs the following steps:
  1. Choose a random $\eta_i \in \mathbb{Z}_q^*$ and compute $r_i = g^{\eta_i}$.
  2. For $1 \leq j \leq t$, compute $f_j = H_2(m_j)$.
  3. For $1 \leq j \leq t$, compute $z_{i,j} = s_i f_j^{\eta_i}$.
  4. Output the batch signature $\sigma_i = (r_i, z_{i,1}, ..., z_{i,t})$.
– Aggregate: Anyone can aggregate a collection of signatures $\{\sigma_i = (r_i, z_{i,1}, ..., z_{i,t})\}_{1 \leq i \leq x}$ on the messages $\{m_j\}_{1 \leq j \leq t}$ from x signers into a batch multi-signature. In particular, $\{\sigma_i = (r_i, z_{i,1}, ..., z_{i,t})\}_{1 \leq i \leq x}$ can be aggregated into $(w, d_1, ..., d_t)$, where

$$w = \prod_{i=1}^{x} r_i, \quad d_j = \prod_{i=1}^{x} z_{i,j}.$$

– BM.Verify: To check the validity of the above batch multi-signature $(w, d_1, ..., d_t)$, the verifier computes $Q = \hat{e}(\prod_{i=1}^{s} H_1(ID_i), g_1)$ and for $1 \leq j \leq t$ checks

$$\hat{e}(d_j, g) \stackrel{?}{=} \hat{e}(f_j, w) \cdot Q.$$

If all the equations hold, the verifier outputs "all valid"; otherwise, it outputs an index set $\mathbb{I}$, which means the multi-signatures with indices in that set are invalid.

The following result relates the security of the IB-B-MS primitive with the difficulty of solving the CDH problem.

Theorem 1. *Suppose an adversary $\mathcal{A}$ who asks at most q_{H_1} times H_1 queries, q_{H_2} times H_2 queries, q_e times* Extract *queries, q_s times* Sign *queries with maximal message size N, and wins the game in Section 4.2 with advantage $\mathsf{Adv}(\mathcal{A})$ in time τ. Then there exists an algorithm to solve the CDH problem with advantage*

$$\frac{4}{(q_e + q_s + x + 1)^2 e^2} \mathsf{Adv}(\mathcal{A})$$

in time $\tau + \mathcal{O}(2q_{H_1} + q_{H_2} + 4Nq_s)\tau_{G_1}$.

Proof. See Appendix A.

5 ID-Based Authenticated Asymmetric Group Key Agreement Protocol

In this section, we propose our IB-AAGKA protocol from bilinear maps.

5.1 The Proposal

In the sequel, we will consider a group of n participants who wish to establish a broadcast channel.

– Setup: The same as BM.Setup, except that an identity-based signature scheme and a cryptographic hash function $H_3 : \mathbb{G}_2 \longrightarrow \{0,1\}^{\varsigma}$ are chosen, where ς is the bit-length of plaintexts.
– Extract: The same as BM.Extract.

– Agreement: A protocol participant $\mathcal{U}_i$, whose identity is ID_i and private key is s_i, performs the following steps:
  1. Choose a random $\eta_i \in \mathbb{Z}_q^*$, compute $r_i = g^{\eta_i}$.
  2. For $1 \leq j \leq n$, compute $f_j = H_2(j)$.
  3. For $1 \leq j \leq n$, compute $z_{i,j} = s_i f_j^{\eta_i}$.
  4. Publish $\sigma_i = (r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\ldots,n\}, j\neq i})$, where ϱ_i is the identity-based signature on r_i. To keep the whole protocol efficient, one may choose an identity-based signature scheme that supports batch verification [12] to generate ϱ_i. The material used to generate the encryption/decryption key is described in Table 1, in which $\underline{z_{i,i}} = z_{i,i}$ is not published, but is kept secret by $\mathcal{U}_i$.

Table 1. Material used to generate the encryption/decryption key

Required for	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$	$\cdots$	$\mathcal{U}_n$	All
$\mathcal{U}_1 \Rightarrow$	$\underline{z_{1,1}}$	$z_{1,2}$	$z_{1,3}$	$\cdots$	$z_{1,n}$	(r_1, ϱ_1)
$\mathcal{U}_2 \Rightarrow$	$z_{2,1}$	$\underline{z_{2,2}}$	$z_{2,3}$	$\cdots$	$z_{2,n}$	(r_2, ϱ_2)
$\mathcal{U}_3 \Rightarrow$	$z_{3,1}$	$z_{3,2}$	$\underline{z_{3,3}}$	$\cdots$	$z_{3,n}$	(r_3, ϱ_3)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathcal{U}_n \Rightarrow$	$z_{n,1}$	$z_{n,2}$	$z_{n,3}$	$\cdots$	$\underline{z_{n,n}}$	(r_n, ϱ_n)
Key	d_1	d_2	d_3	$\cdots$	d_n	(w, Q)

– Enc.Key.Gen: To get the group encryption key, an entity first checks the n message-signature pairs $(r_1, \varrho_1), ..., (r_n, \varrho_n)$. If all of these signatures are valid, then the entity computes

$$w = \prod_{i=1}^{n} r_i, Q = \hat{e}(\prod_{i=1}^{n} H_1(ID_i), g_1),$$

and sets the group encryption key as (w, Q).

– Dec.Key.Gen: Each participant $\mathcal{U}_i$ checks the n message-signature pairs $(r_1, \varrho_1), ..., (r_n, \varrho_n)$. If all of these signatures are valid, $\mathcal{U}_i$ computes $d_i = \prod_{j=1}^{n} z_{j,i}$, and checks

$$\hat{e}(d_i, g) \stackrel{?}{=} \hat{e}(f_i, w) \cdot Q.$$

If the equation holds, $\mathcal{U}_i$ accepts d_i as the group decryption key; otherwise, it aborts.

– Enc: For a plaintext m, an entity[5] generates the ciphertext by the following steps:
  1. Select $\rho \in \mathbb{Z}_q^*$, compute $c_1 = g^\rho, c_2 = w^\rho, c_3 = m \oplus H_3(Q^\rho)$.
  2. Output the ciphertext $c = (c_1, c_2, c_3)$.

[5] Unlike the conventional GKA protocols, not only the protocol participants can send a ciphertext to the group, but also any outsider who learns the group encryption key.

– Dec: To decrypt the ciphertext $c = (c_1, c_2, c_3), \mathcal{U}_i$, whose group decryption key is d_i, computes

$$m = c_3 \oplus H_3(\hat{e}(d_i, c_1)\hat{e}(f_i^{-1}, c_2)).$$

Theorem 2. *Suppose an adversary $\mathcal{A}$ who asks at most q_{H_1} times H_1 queries, q_{H_2} times H_2 queries, q_{H_3} times H_3 queries, q_c times* Corrupt *queries, q_s times* Send *queries, q_{er} times* Ek.Reveal *queries and q_{dr} times* Dk.Reveal *queries, and wins the game with advantage* $\mathsf{Adv}(\mathcal{A})$ *in time τ. Then there exists an algorithm to solve the k-BDHE problem with advantage*

$$\frac{4(1 - k\mathsf{Adv}_{sig}(\mathcal{A}))}{q_{H_3}(q_c + q_{dr} + k + 1)^2 e^2}\mathsf{Adv}(\mathcal{A}).$$

in time $\tau + \mathcal{O}(q_{er})\tau_{\hat{e}} + \mathcal{O}(2q_{H_1} + q_{H_2} + kq_s)\tau_{G_1}$, where $\mathsf{Adv}_{sig}(\mathcal{A})$ is the advantage for $\mathcal{A}$ to forge a valid identity-based signature in time τ, $\tau_{\hat{e}}$ is the time to compute a pairing and τ_{G_1} is the time to compute a scalar multiplication in $\mathbb{G}_1$.

Proof. See Appendix B.

6 Conclusion

We have defined a security model for IB-AAGKA protocols and proposed a one-round IB-AAGKA protocol from bilinear maps based on the k-BDHE assumption in the random oracle model. The new protocol allows an adversary to adaptively choose his targets, and it offers the key secrecy, known-key security and partial forward secrecy properties. This design is also readily adaptable to provide broadcast encryption [7,15].

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange. In: STOC 1998, pp. 419-428. ACM Press, New York (1998)
2. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Goos, G. and Hartmanis, J. (eds.) CRYPTO 1993. LNCS, vol. 773, pp. 232-249. Springer, Heidelberg (1994)
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Proc. of the First ACM Conference on Computer and Communications Security, pp. 62-73 (1993)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure Against Dictionary Attacks,. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139-155. Springer, Heidelberg (2000)
5. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440-456. Springer, Heidelberg (2005)
6. Boneh, D., Franklin, M.: Identity Based Encryption from the Weil Pairing. SIAM J. of Computing 32(3), 586-615 (2003)
7. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455-470. Springer, Heidelberg (2008)
8. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie - Hellman Key Exchange - The Dynamic Case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290-309. Springer, Heidelberg (2001)
9. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321-336. Springer, Heidelberg (2002)
10. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: Provably Authenticated Group Diffie-Hellman Key Exchange. In: Samarati, P. (ed.) ACM CCS 2001, pp. 255-264. ACM Press, New York (2001)
11. Burmester, M., Desmedt, Y.G.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275-286. Springer, Heidelberg (1995)

12. Camenisch, J., Hohenberger, S., Pedersen, M.: Batch Verification of Short Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246-263. Springer, Heidelberg (2007)
13. Choi, K.Y., Hwang, J.Y., Lee, D.H.: Efficient ID-Based Group Key Agreement with Bilinear Maps. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 130-144. Springer, Heidelberg (2004)
14. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644-654 (1976)
15. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480-491. Springer, Heidelberg (1994)
16. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537-554. Springer, Heidelberg (1999)
17. Gentry, C., Ramzan, Z.: Identity-Based Aggregate Signatures. In: Yung, M., et al. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257-273. Springer, Heidelberg (2006)
18. Gorantla, M.C., Boyd, C., Nieto, J.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 105-123. Springer, Heidelberg (2009)
19. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. J. of Cryptology 17, 263-276 (2004)
20. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110-125. Springer, Heidelberg (2003)
21. Katz, J., Shin, J.S.: Modeling Insider Attacks on Group Key-Exchange Protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security-CCS 2005, pp. 180-189. ACM, New York (2005)
22. Kim, H., Lee, S., Lee, D.H.,: Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245-259. Springer, Heidelberg (2004)
23. Kudla, C., Paterson, K.G.: Modular Security Proofs for Key Agreement Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549-565. Springer, Heidelberg (2005)
24. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47-53. Springer (1985)
25. Wu, Q., Mu, Y., Susilo, W., Qin, B., Domingo-Ferrer, J.: Asymmetric Group Key Agreement. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 153-170. Springer, Heidelberg (2009)

A Proof of Theorem 1

Proof. Let $\mathcal{C}$ be a challenger, $\mathcal{A}$ be an adversary who can break the proposed IB-B-MS scheme under an adaptive chosen-message attack. Suppose that $\mathcal{C}$ is given an instance $(g, g^{\alpha}, g^{\beta})$ of the CDH problem in G_1. We show how $\mathcal{C}$ can use $\mathcal{A}$ to solve the CDH problem, *i.e.*, to compute $g^{\alpha\beta}$.

Initial: Firstly, $\mathcal{C}$ sets $g_1 = g^{\alpha}$, then selects the system parameters $\mathsf{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, H_1, H_2)$, and gives params to $\mathcal{A}$. In the following, we treat H_1 and H_2 as random oracles which are controlled by $\mathcal{C}$.

Training: $\mathcal{C}$ answers $\mathcal{A}$'s queries as follows:

H_1 queries: $\mathcal{C}$ maintains an initially empty list H_1^{list}. On input ID_i, $\mathcal{C}$ does the following

- If there is a tuple $(ID_i, \mu_i, id_i, s_i, H_1coin_i)$ on H_1^{list}, return id_i as the answer.
- Else flip a coin $H_1coin_i \in \{0, 1\}$ that yields 1 with probability δ and 0 with probability $1-\delta$, pick a random $\mu_i \in Z_q^*$ and proceed as follows
  - If $H_1coin_i = 0$, set $id_i = g^{\mu_i}, s_i = g_1^{\mu_i}$, add $(ID_i, \mu_i, id_i, s_i, H_1coin_i)$ to H_1^{list} and respond with id_i.
  - Else set $id_i = g^{\beta\mu_i}, s_i = null$, add $(ID_i, \mu_i, id_i, s_i, H_1coin_i)$ to H_1^{list} and respond with id_i.

H_2 queries: $\mathcal{C}$ keeps an initially empty list H_2^{list}. On input m_i, $\mathcal{C}$ does the following:

- If there is a tuple $(m_i, \nu_i, f_i, H_2coin_i)$ on H_2^{list}, return f_i as the answer.

– Else flip a coin H_2coin_i that yields 1 with probability δ and 0 with probability $1-\delta$, randomly select $\beta_i \in Z_q^*$ and do the following
  - If $H_2coin_i = 0$, set $f_i = g^{\nu_i} g^{\alpha}$, add $(m_i, \nu_i, f_i, H_2coin_i)$ to H_2^{list} and return f_i as the answer.
  - Else compute $f_i = g^{\nu_i}$, add $(m_i, \nu_i, f_i, H_2coin_i)$ to H_2^{list} and return f_i as the answer.

Extract queries: On input an identity ID_i, $\mathcal{C}$ first makes an H_1 query on ID_i, then recovers $(ID_i, \mu_i, id_i, s_i, H_1coin_i)$ from H_1^{list}. If $H_1coin_i = 0$, $\mathcal{C}$ returns s_i as the answer; otherwise, it aborts.

Sign queries: On input $(ID_i, m_1, ..., m_n)$ with $n \leq N$, $\mathcal{C}$ first asks an H_1 query on ID_i and finds $(ID_i, \mu_i, id_i, s_i, H_1coin_i)$ on H_1^{list}, and for $1 \leq j \leq n$, asks an H_2 on m_j and recovers $(m_j, \nu_j, f_j, H_2coin_j)$ from H_2^{list}; then $\mathcal{C}$ does the following:

– If $H_1coin_i = 0$, use the Sign algorithm to generate a batch signature.
– Else if $H_2coin_j = 0$ for all $1 \leq j \leq n$, select $\eta_i \in Z_q^*$, compute $r_i = g^{\eta_i} g^{-\beta\mu_i}$, $z_{i,j} = r_i^{\nu_j} g_1^{\eta_i}$, and output $(r_i, z_{i,1}, ..., z_{i,n})$.
– Else abort.

Forgery: Finally, $\mathcal{A}$ outputs an identity set $\mathbb{L}_{ID}^* = \{ID_1^*, ..., ID_x^*\}$, a message m^* and a multi-signature $\sigma^* = (w^*, d^*)$, where σ^* is a valid multi-signature on m^* under $(ID_1^*, ..., ID_x^*)$.

In order to get the solution of the CDH problem, $\mathcal{C}$ now proceeds with the following steps:

1. For $1 \leq i \leq x$, ask an H_1 query on ID_i^*, and recover $(ID_i^*, \mu_i^*, id_i^*, s_i, H_1coin_i^*)$ from H_1^{list}.
2. Ask an H_2 query on m^*, and recover $(m^*, \nu^*, f^*, H_2coin^*)$ from H_2^{list}.

In the following, for simplicity, we will only consider the case that $H_1coin_1^* = H_2coin^* = 1$ and $H_1coin_i^* = 0$ for $2 \leq i \leq t$; otherwise, $\mathcal{C}$ aborts. If $\mathcal{C}$ does not abort, this implies $id_1^* = g^{\beta\mu_1^*}$, $f^* = g^{\nu^*}$, and for $2 \leq i \leq t$, $id_i^* = g^{\mu_i^*}$. Since σ^* should be valid, we have

$$\hat{e}(d^*, g) = \hat{e}(f^*, w^*)\hat{e}(\prod_{i=1}^{t} H_1(ID_i^*), g_1).$$

It is easy to get

$$g^{\alpha\beta} = (d^* \cdot w^{*-\nu^*} \cdot g_1^{-\sum_{i=2}^{t} \mu_i^*})^{\mu_1^{*-1}}$$

as the solution of CDH problem.

To complete the proof, it remains to compute the probability that $\mathcal{C}$ solves the given instance of the CDH problem. First, we analyze the three events needed for $\mathcal{C}$ to succeed:

– $\mathcal{E}1$: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$'s queries.
– $\mathcal{E}2$: σ^* is a valid and nontrivial multi-signature on m^* under $(ID_1^*, ..., ID_x^*)$.
– $\mathcal{E}3$: Event $\mathcal{E}2$ occurs, and also $H_1coin_1^* = H_2coin^* = 1$ and for $2 \leq i \leq x$ $H_1coin_i^* = 0$.

$\mathcal{C}$ succeeds if all of these events happen. The probability $\Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3]$ can be decomposed as

$$\Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3] = \Pr[\mathcal{E}1] \Pr[\mathcal{E}2|\mathcal{E}1] \Pr[\mathcal{E}3|\mathcal{E}1 \wedge \mathcal{E}2].$$

From the above simulation, it is easy to get $\Pr[\mathcal{E}1] \geq (1-\delta)^{q_e+q_s}$, $\Pr[\mathcal{E}2|\mathcal{E}1] \geq \mathsf{Adv}(A)$ and $\Pr[\mathcal{E}3|\mathcal{E}1 \wedge \mathcal{E}2] = \delta^2(1-\delta)^{x-1}$. Hence, we have

$$\begin{aligned}\Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3] &\geq \delta^2(1-\delta)^{q_e+q_s+x-1}\mathsf{Adv}(A) \\ &\geq \frac{4}{(q_e+q_s+x+1)^2e^2}\mathsf{Adv}(A).\end{aligned}$$

The time complexity is

$$\tau + \mathcal{O}(2q_{H_1} + q_{H_2} + 4Nq_s)\tau_{G_1}.$$

□

B Proof of Theorem 2

Proof. Let $\mathcal{C}$ be a challenger, and $\mathcal{A}$ be an adversary who can break the proposed protocol. Suppose $\mathcal{C}$ is given an instance $(g, h, y_1, ..., y_k, y_{k+2}, ..., y_{2k})$ of the k-BDHE problem, where $y_i = g^{\alpha^i}, i \in \{1, ..., k, k+2, ..., 2k\}$ with some unknown $\alpha \in \mathbb{Z}_q^*$. We show how $\mathcal{C}$ can use $\mathcal{A}$ to solve the problem, *i.e.*, to compute the required $\hat{e}(g,h)^{\alpha_{k+1}}$.

In the following, we assume that in each session the participants involved in the group are at most of scale k. As defined in Section 3.1, when a new session is first initiated, it will have a unique session ID. We assume that the session ID will be sid_ι. At the same time, $\mathcal{C}$ will also flip a coin c_{sid_ι} so that $\Pr[c_{\mathsf{sid}_\iota} = 1] = \delta, \Pr[c_{\mathsf{sid}_\iota} = 0] = 1-\delta$. The tuple $(\mathsf{sid}_\iota, c_{\mathsf{sid}_\iota})$ is recorded by $\mathcal{C}$.

Initial: When the game begins, $\mathcal{C}$ selects the system parameters $\mathsf{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, \varsigma, H_1, H_2, H_3, sig)$, where $g_1 = y_1 = g^\alpha$, sig is a secure identity-based signature scheme. params is passed to $\mathcal{A}$. In the following, we treat H_1, H_2 and H_3 as random oracles which are controlled by $\mathcal{C}$.

Training: $\mathcal{C}$ answers $\mathcal{A}$'s queries as follows:

H_1 queries: $\mathcal{C}$ maintains an initially empty list H_1^{list}. On input ID_i, $\mathcal{C}$ does the following:

- If ID_i already appears on the H_1^{list} in a tuple $(ID_i, \mu_i, id_i, s_i, c_{H_1^i})$, return id_i as the answer.
- Else, pick a random $\mu_i \in \mathbb{Z}_q^*$, generate a random coin $c_{H_1^i}$ so that $\Pr[c_{H_1^i} = 1] = \delta, \Pr[c_{H_1^i} = 0] = 1-\delta$ and proceed as follows:
  - If $c_{H_1^i} = 0$, set $id_i = g^{\mu_i}, s_i = g_1^{\mu_i}$, add $(ID_i, \mu_i, id_i, s_i, c_{H_1^i})$ to H_1^{list} and respond with id_i.
  - Else, set $id_i = g^{\mu_i}y_k, s_i = null$, add $(ID_i, \mu_i, id_i, s_i, c_{H_1^i})$ to H_1^{list} and respond with id_i.

H_2 queries: $\mathcal{C}$ keeps an initially empty list H_2^{list}. On input j, $\mathcal{C}$ does the following:

- If there is a tuple (j, ν_j, f_j) on H_2^{list}, return f_j as the answer.
- Else if $j \leq k$, randomly select $\nu_j \in \mathbb{Z}_q^*$, set $f_j = y_j g^{\nu_j}$, add (j, ν_j, f_j) to H_2^{list} and return f_j as the answer.
- Else, randomly select $\nu_j \in \mathbb{Z}_q^*$, set $f_j = g^{\nu_j}$, add (j, ν_j, f_j) to H_2^{list} and return f_j as the answer.

H_3 queries: $\mathcal{C}$ maintains an initially empty list H_3^{list}. On input a message Ω_i, if there is a tuple (Ω_i, ϖ_i) on H_3^{list}, $\mathcal{C}$ returns ϖ_i as the answer; otherwise, $\mathcal{C}$ randomly selects $\varpi_i \in \{0,1\}^\varsigma$, adds (Ω_i, ϖ_i) to H_3^{list} and responds with ϖ_i.

$\mathsf{Corrupt}(\mathcal{U}_i)$: Suppose the identity of $\mathcal{U}_i$ is ID_i. On receiving the corrupt query, $\mathcal{C}$ first submits ID_i to the H_1 oracle if this query has never been asked before, recovers $(ID_i, \mu_i, id_i, s_i, c_{H_1^i})$ on H_1^{list}, and does the following:

– If $c_{H_1^i} = 1$, abort (Event 1).
– Otherwise, return s_i as the answer.

$\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Delta)$: $\mathcal{C}$ maintains an initially empty list S^{list}. Assume $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \{ID_1, ..., ID_n\}$. To answer this query, $\mathcal{C}$ first recovers $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}}$ corresponding to $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$, submits ID_i to the H_1 oracle if this query has never been asked before, recovers $(ID_i, \mu_i, id_i, s_i, c_{H_1^i})$ from H_1^{list}, submits j to the H_2 oracle and recovers (j, ν_j, f_j) from H_2^{list} for $1 \leq j \leq n$; then $\mathcal{C}$ simulates this oracle as follows:

– If $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 0$ and $c_{H_1^i} = 0$, simulate this query normally:
  1. Choose a random $\eta_i \in \mathbb{Z}_q^*$ and compute $r_i = g^{\eta_i}$.
  2. For $0 \leq j \leq n$, compute $z_{i,j} = s_i f_j^{\eta_i}$.
  3. Add $(ID_i, \mathsf{sid}^{\pi}_{\mathcal{U}_i}, \eta_i, z_{i,i})$ to S^{list} and publish $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,...,n\}, j\neq i})$.
– Else if $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 0$ and $c_{H_1^i} = 1$, generate the answer as follows:
  1. Select $\eta_i \in \mathbb{Z}_q^*$ and compute $r_i = g^{\eta_i} \prod_{l=1}^{n} y_{k-l+1}^{-1}$,
  2. For $1 \leq j \leq n$, $z_{i,j} = f_j^{\eta_i} g_1^{\mu_i} \prod_{l=1}^{n, l\neq j} y_{k-l+1+j}^{-1}$.
  3. Add $(ID_i, \mathsf{sid}^{\pi}_{\mathcal{U}_i}, \eta_i, z_{i,i})$ to S^{list} and publish $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,...,n\}, j\neq i})$.
– Else if $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 1$ and $c_{H_1^i} = 0$, perform the following steps:
  1. Select $\eta_i \in \mathbb{Z}_q^*$ at random and compute $r_i = g_1^{\eta_i} y_{k-i+1}$.
  2. For $1 \leq j \leq n, i \neq j$, compute $z_{i,j} = r_i^{\nu_j} g_1^{\mu_i} y_j^{\eta_i} y_{k-i+1+j}$.
  3. Add $(ID_i, \mathsf{sid}^{\pi}_{\mathcal{U}_i}, \eta_i, null)$ to S^{list} and respond with $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,...,n\}, j\neq i})$.
– Else, $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 1$ and $c_{H_1^i} = 1$, do the following:
  1. Randomly select $\eta_i \in \mathbb{Z}_q^*$ and compute $r_i = g_1^{\eta_i} \prod_{l=1}^{n, l\neq i} y_{k-l+1}^{-1}$.
  2. For $1 \leq j \leq n, i \neq j$, compute $z_{i,j} = r_i^{\nu_j} g_1^{\mu_i} y_j^{\eta_i} \prod_{l=1}^{n, l\notin\{i,j\}} y_{k-l+1+j}^{-1}$.
  3. Add $(ID_i, \mathsf{sid}^{\pi}_{\mathcal{U}_i}, \eta_i, null)$ to S^{list} and respond with $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,...,n\}, j\neq i})$.

$\mathsf{Ek.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: Assume $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma_1, ..., \sigma_n\}$, where $\sigma_l = (r_l, \varrho_l, \{z_{l,j}\}_{j\in\{1,...,n\}, j\neq l})$. If $\mathsf{state}^{\pi}_{\mathcal{U}_i} = successfully\ terminated$, $\mathcal{C}$ computes $w = \prod_{l=1}^{n} r_l$, $Q = \hat{e}(\prod_{l=1}^{n} H_1(ID_l), g_1)$, and returns (w, Q); otherwise, it returns $null$.

$\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: $\mathcal{C}$ first finds $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}}$ corresponding to $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$ and then does the following:

– If $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 1$, abort (Event 2).
– Else if $\mathcal{U}_i{}^{\pi}$ is not *successfully terminated*, return $null$.
– Else, recover the corresponding $z_{i,i}$ from S^{list} and $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma_1, ..., \sigma_n\}$, where $\sigma_l = (r_l, \varrho_l, \{z_{l,j}\}_{j\in\{1,...,n\}, j\neq l})$; compute and output $d_i = \prod_{l=1}^{n} z_{l,i}$.

$\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: At some point, $\mathcal{A}$ chooses a fresh $\Pi^{\pi}_{\mathcal{U}_i}$ and two messages m_0, m_1 on which it wishes to be challenged. Assume $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \{ID_1^*, ..., ID_n^*\}$, $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma_1^*, ..., \sigma_n^*\}$, where $\sigma_l^* = (r_l^*, \varrho_l^*, \{z_{l,j}^*\}_{j\in\{1,...,n\}, j\neq l})$ and ϱ_l^* is a valid signature on r_l^*. $\mathcal{C}$ does the following:

1. For $1 \leq l \leq n$, recover $(ID_l^*, \mu_l^*, id_l^*, s_l^*, c_{H_1^l}^*)$ from H_1^{list}.
2. If $c_{\mathsf{sid}^{\pi}_{\mathcal{U}_i}} = 1$, and, there exists one and only one $c_{H_1^l}^* = 1$, and, $\{\varrho_l^*\}_{l\in\{1,...,n\}}$ is not a forgery, turn to Step 3. Otherwise, abort (Event 3).

3. For $1 \leq l \leq n$, recover $\eta_1^*, ..., \eta_n^*$ from S^{list}, where η_l^* is the random value chosen to generate r_l^*. Note that, since ϱ_l^* is a valid signature on r_l^*, $\mathsf{ek}_{\mathcal{U}_i}^{\pi} = (w^*, Q^*)$ equals $(g^{\sum_{l=1}^{n} \eta_l^*}, g^{\sum_{l=1}^{n} \mu_l^*} y_k)$.
4. Set $c_1 = h, c_2 = h^{\sum_{l=1}^{n} \eta_l^*}$, randomly choose $\theta \in \{0,1\}^{\varsigma}$, and compute $c_3 = m_b \oplus \theta$, where $b \in \{0,1\}$.
5. Return $c = (c_1, c_2, c_3)$. Note that $\mathcal{A}$ cannot recognize that c is not a proper ciphertext unless she queries H_3 on $\hat{e}(g_1^{\sum_{l=1}^{n} \mu_l^*} g^{\alpha^{k+1}}, h)$.

Response: Once $\mathcal{A}$ finishes querying and returns its guess $b' \in \{0,1\}$, $\mathcal{C}$ randomly chooses a tuple (Ω_i, ϖ_i) from H_3^{list} and returns the value $\Omega_i \cdot \hat{e}(g_1^{-\sum_{l=1}^{n} \mu_l^*}, h)$ as the response to the k-BDHE challenge.

We note that the above simulations of all the random oracles are valid and the messages of the oracles are uniformly distributed in the message space. Hence, the adversary cannot find inconsistence between the simulation and the real world. Therefore, $\Pr[b = b'] \geq \mathsf{Adv}(\mathcal{A})$. It remains to determine the probability that $\mathcal{C}$ outputs the required Ω_i. It is easy to see that $\mathcal{C}$ will abort if Event 1 or Event 2 or Event 3 happens. We must calculate $\Pr[\neg\text{Event 1} \wedge \neg\text{Event 2} \wedge \neg\text{Event 3}]$.

By our setting, it is easy to get $\Pr[\neg\text{Event 1}] \geq (1-\delta)^{q_c}$, $\Pr[\neg\text{Event 2}] \geq (1-\delta)^{q_{dr}}$, $\Pr[\neg\text{Event 3}] \geq \delta^2(1-\delta)^{n-1}(1 - n\mathsf{Adv}_{sig}(\mathcal{A}))$. Since these probabilities are independent, the overall probability that $\mathcal{C}$ does not abort is

$$\delta^2(1-\delta)^{q_c+q_{dr}+n-1}(1 - n\mathsf{Adv}_{sig}(\mathcal{A})) \geq \delta^2(1-\delta)^{q_c+q_{dr}+k-1}(1 - k\mathsf{Adv}_{sig}(\mathcal{A})).$$

This value is maximized at $\delta = \frac{2}{q_c+q_{dr}+k+1}$. Hence, we have

$$\Pr[\neg\text{Event 1} \wedge \neg\text{Event 2} \wedge \neg\text{Event 3}] \geq \frac{4(1 - k\mathsf{Adv}_{sig}(\mathcal{A}))}{(q_c + q_{dr} + k + 1)^2 e^2}.$$

In conclusion, we have the probability for $\mathcal{C}$ to solve the k-BDHE problem is

$$\frac{4(1 - k\mathsf{Adv}_{sig}(\mathcal{A}))}{q_{H_3}(q_c + q_{dr} + k + 1)^2 e^2}\mathsf{Adv}(\mathcal{A}).$$

The time complexity is

$$\tau + \mathcal{O}(q_{er})\tau_{\hat{e}} + \mathcal{O}(2q_{H_1} + q_{H_2} + kq_s)\tau_{G_1}.$$

□