Provably Secure One-Round Identity-Based Authenticated Asymmetric Group Key Agreement Protocol ☆

Lei Zhang[a,b,*], Qianhong Wu[b,c], Bo Qin[b,d], Josep Domingo-Ferrer[b]

[a] *East China Normal University, Software Engineering Institute, Shanghai, China*
[b] *Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics UNESCO Chair in Data Privacy, Tarragona, Catalonia*
[c] *Wuhan University, School of Computer Key Lab. of Aerospace Information Security and Trusted Computing Ministry of Education, China*
[d] *Xi'an University of Technology, School of Science, Department of Maths, China*
leizhang@sei.ecnu.edu.cn,{qianhong.wu,bo.qin,josep.domingo}@urv.cat

Abstract

The newly introduced notion of *asymmetric group key agreement* (AGKA) enables external users to securely send messages to group members. The existing AGKA is only secure against passive attacks which are too weak to capture the attacks in the real world. In this paper, we formalize an active security model for identity-based authenticated asymmetric group key agreement (IB-AAGKA). We then present an efficient identity-based batch multi-signature, from which we construct an IB-AAGKA protocol. Our protocol is proven secure under the Bilinear Diffie-Hellman Exponent (BDHE) assumption. The active security feature implies that the protocol can withstand more realistic attacks. The identity-based feature eliminates the need of certificates and solves the certificate management problem in traditional public-key cryptosystems. Finally, an effective trade-off is provided to balance the protocol transcript size and the ciphertext size.

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

☆A preliminary version of this paper appeared in [27].
*Corresponding author.

1. Introduction

Group Key Agreement (GKA) protocols are mainly used to implement secure channels in group-oriented communications. They are widely employed in many collaborative and distributed applications such as multi-party computations, file distribution/sharing, audio/video conference and chat systems. Conventionally, in a GKA protocol, a group of members interact over an open network to establish a common secret key. This secret key is then used to encrypt any message to group members. Since the common secret key is solely known to the group members, only a group member can send secret messages to other group members. In the real world, however, anyone is a potential sender, just as in public-key encryption. Observing this fact, Wu *et al.* introduced a notion called Asymmetric Group Key Agreement (AGKA,[26]). In their definition, instead of a common secret key, the group members negotiate a common encryption key which is publicly accessible to any entity, but they hold respective secret decryption keys.

1.1. Motivation and Contributions

With the standard round notion, existing GKA protocols require two or more rounds to establish a secret session key. As remarked in [20, 26], one-round key agreement protocols have several advantages over key agreement protocols in two or more rounds. For instance, imagine a group of friends who wish to share their personal documents via an open network. If a two-round key agreement protocol is employed, all group members should be online at the same time. However, if group members live in different time zones, it is very inconvenient to require them to be online concurrently.

A trivial way to realize one-round AGKA is to let each member publish a public key and withhold the respective secret key. To send a message to this group, a sender separately encrypts to each member and generates the final ciphertext by concatenating all the underlying individual ones. This trivial solution leads to $\mathcal{O}(n)$-size ciphertext and requires $\mathcal{O}(n)$ encryption operations for a group of n receivers. The challenge is to design AGKA allowing short ciphertext and efficient encryption. To this end, Wu *et al.* [26] proposed a non-trivial 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. Their protocol requires $\mathcal{O}(1)$-size ciphertext and $\mathcal{O}(1)$ encryption operations after the group encryption key is negotiated.

Though Wu *et al.*'s protocol is efficient, it is only secure against passive adversaries who just eavesdrop the communication channel. In the real world, the adversary is more likely to be active and may control the communication channel 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 the well-known man-in-the-middle attack. Hence, it is vital for an AGKA protocol to withstand active adversaries. This calls for authenticated key agreement protocols.

An authenticated key agreement protocol [12, 13, 19, 29] ensures that no entities aside from the intended ones can possibly compute the agreed session key, even if the attacker is active. Authenticated key agreement protocols may be designed under different public-key cryptosystems. A number of key agreement protocols have been proposed in the traditional public-key infrastructure setting. In that paradigm, users obtain each other's certificate, extract each other's public key, check their certificate chains and finally generate a shared session key. Consequently, the management of the certificates incurs a heavy burden on computation, storage, and communication. To eliminate such costs, Identity-Based Public Key Cryptography (IB-PKC) was introduced by Shamir [25] in 1984. The distinguishing 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 formalize a security model for Identity-Based Authenticated Asymmetric Group Key Agreement (IB-AAGKA) protocols. We first define the basic security in IB-AAGKA against indistinguishable chosen identity and plaintext attacks (Ind-ID-CPA). Then we strengthen the security to capture more powerful attacks called indistinguishable chosen identity and ciphertext attacks (Ind-ID-CCA), in which the attacker can also access decryption oracles to obtain the messages encrypted under the negotiated group public key. Both models allow an adversary to adaptively choose his targets to collude, and capture the identity based variation of the typical security requirements (*i.e.*, *secrecy*, *known-key security* and *forward secrecy*) in GKA protocols. The security requirements are understood as follows:

- Secrecy requires that only the legitimate participants (group members) can read the messages encrypted under 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 the earlier runs of the protocol. Specifically, we say that a key agreement protocol offers *perfect forward secrecy* if compromising the long-term private keys of all the group members does not threaten any previously established group decryption key. We say that a key agreement offers *partial forward secrecy* if compromising the long-term private keys of one or more specific group members does not threaten the group decryption keys previously established by these group members.

In the preliminary version [27] of this paper , we proposed a non-trivial one-round IB-AAGKA protocol which is secure against Ind-ID-CPA. In this paper, detailed proofs are provided in the random oracle model [3] for the security results in [27]. Further, an enhanced IB-AAGKA protocol is proposed that achieves stronger security against Ind-ID-CCA. In our protocols, each participant needs to publish $\mathcal{O}(n)$ group elements during the protocol execution and the ciphertext is of size $\mathcal{O}(1)$. We then suggest an efficient tradeoff to achieve a balance between the ciphertext size and the protocol transcript size per member, so that both sizes are $\mathcal{O}(\sqrt{n})$. Finally, we show that our scheme is secure against some insider attacks.

Our protocols are built on a new notion referred to as identity-based batch multi-signature (IB-B-MS), which may be of independent interest. Our IB-B-MS scheme allows multiple signers to sign t messages in an efficient way. The batch multi-signature consists of only $t + 1$ group elements, independently of the number of signers. Furthermore, the batch multi-signature can be separated into t individual multi-signatures, which seems an interesting property to build advanced systems for some applications.

1.2. Related Work

Since Diffie and Hellman published the first solution to key agreement [15], much attention has been paid to this primitive. Joux [20] was the first to extend key agreement to three parties without introducing extra rounds. Both the Diffie-Hellman and the Joux protocols are one-round. However, when there are more than three participants, it seems knotty to construct key agreement protocols without additional rounds. Indeed, many attempts have been made to extend the Diffie-Hellman and Joux protocols to n parties. Among them, the Burmester-Desmedt protocol [10] needs two rounds and is the most efficient GKA protocol in round efficiency without constraints on n.

For a key agreement protocol to be deployed in open networks, it should be secure against active adversaries. However, the basic Diffie-Hellman and Joux protocols as well as the Burmester-Desmedt protocol do not authenticate the communicating 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 [14, 23, 24]; among them, the GKA protocol in [14] is based on IB-PKC. This protocol refers to Katz and Yung's results [21] for an authenticated version and requires two rounds.

The paradigm of provable security subsumes a formalization that considers the protocol environment and identifies its security goals. Bresson *et al.* [9] were the first to formalize the security model for group key agreement protocols. Their model extends the previous security model for key agreement protocols between two or three parties [1, 2, 4]. Subsequently, this model was refined by Bresson *et al.* [7, 8] and modified for specific properties in [21, 22]. These models are widely accepted for proving the security of GKA protocols. In this paper, we will extend these models to define the security of 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 presented in Section 4. Section 5 proposes and analyzes our IB-AAGKA protocols. In Section 6, we show that our protocols are also secure against some insider attacks. Finally, we conclude in Section 7.

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 briefly reviewed next.

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 states 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 $\hat{e}(g,h)^{\alpha^{k+1}}$. Since the input vector lacks the term $g^{\alpha^{k+1}}$, the bilinear map does not seem helpful to compute $\hat{e}(g,h)^{\alpha^{k+1}}$.

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

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

in solving k-BDHE problem. The k-BDHE assumption states 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.* [26], derived from the security model for conventional GKA protocols [9]. We note that the security model in [26] 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 a polynomially bounded number 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.

- $\mathsf{sid}_{\mathcal{U}_i}^{\pi}$ is the *session ID* of instance $\Pi_{\mathcal{U}_i}^{\pi}$. We follow [22] and assume 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.

- $\mathsf{ms}_{\mathcal{U}_i}^{\pi}$ is the concatenation of all messages sent and received by $\Pi_{\mathcal{U}_i}^{\pi}$ during its execution, where the messages are ordered by round, and within each round lexicographically by the identities of the purported senders.

- $\mathsf{ek}_{\mathcal{U}_i}^{\pi}$ is the encryption key held by $\Pi_{\mathcal{U}_i}^{\pi}$.

- $\mathsf{dk}_{\mathcal{U}_i}^{\pi}$ is the decryption key held by $\Pi_{\mathcal{U}_i}^{\pi}$.

- $\mathsf{state}_{\mathcal{U}_i}^{\pi}$ represents the current (internal) state of instance $\Pi_{\mathcal{U}_i}^{\pi}$. When an instance has *terminated*, it finishes sending and receiving messages. We say that an IB-AAGKA protocol has been *successfully terminated* (accepted) in the instance $\Pi_{\mathcal{U}_i}^{\pi}$ if it possesses $\mathsf{ek}_{\mathcal{U}_i}^{\pi}(\neq null)$, $\mathsf{dk}_{\mathcal{U}_i}^{\pi}(\neq null)$, $\mathsf{pid}_{\mathcal{U}_i}^{\pi}$ and $\mathsf{sid}_{\mathcal{U}_i}^{\pi}$.

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

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 to establish a confidential channel among users. Hence, we directly use the confidentiality of the final confidential 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 in what 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:

- $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Delta)$[1]: 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.*

- $\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: 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). The challenger 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 $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Delta)$.*

[1]Some models allow the adversary to make $\mathsf{Execute}$ queries. This feature is used to model passive attacks, where the adversary eavesdrops on a 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 [21], 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.

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.

The above model considers security against Ind-ID-CPA. A stronger attack against IB-ASGKA protocols is called indistinguishable chosen identity and ciphertext attack (Ind-ID-CCA), in which an adversary can ask the challenger to decrypt ciphertexts for him. With a slight modification, the above model can capture this attack. That is, during the **Training** stage, $\mathcal{A}$ can additionally submit ciphertexts to $\mathcal{C}$ to obtain the corresponding plaintexts, with the restriction that in the Test query, the challenging ciphertext has never been queried for the plaintext. In this paper, we will mainly focus on the design of an IB-ASGKA protocol secure against Ind-ID-CPA.

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. In 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][2].

[2] To the best of our knowledge, the multi-signature scheme due to Gentry and Ramzan [17] is the most efficient multi-signature scheme from bilinear maps. If only one message is to be signed, our scheme degenerates into the Gentry-Ramzan multi-signature scheme. If the number of the messages to be signed is $n \geq 1$, then the length of our signature is about a half of the Gentry-Ramzan signature, and, at the Sign stage, our scheme can save $n - 1$ exponentiation operations.

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 adaptive chosen-message attacks 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} queries to H_1, q_{H_2} queries to H_2, q_e queries to* Extract*, q_s queries to* Sign *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 $\tau + \mathcal{O}(2q_{H_1} + q_{H_2} + 4Nq_s)\tau_{G_1}$ where τ_{G_1} is the time to compute a scalar multiplication in G_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 $\mathbb{G}_1$. 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 query 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 obtain 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 the 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

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

□

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 securely receive messages from a sender who may be a participant or not.

- 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 $\eta_i \in \mathbb{Z}_q^*$ and 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 [11, 28] to generate ϱ_i. The material used to generate the encryption/decryption key is described in Table 1, in which $\underline{z_{i,i}} = s_i f_i^{\eta_i}$ is not published, but is kept secret by $\mathcal{U}_i$. The elements $\{z_{i,j}\}_{i\in\{1,\ldots,n\}}$ in each column are used to generate $\mathcal{U}_j$'s group decryption key d_j. $\{(r_i, ID_i)\}_{i\in\{1,\ldots,n\}}$ are used to generate the group encryption key (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 above equation holds, $\mathcal{U}_i$ accepts d_i as the group decryption key; otherwise, it aborts.

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, ID_1, ϱ_1)
$\mathcal{U}_2 \Rightarrow$	$z_{2,1}$	$\underline{z_{2,2}}$	$z_{2,3}$	$\cdots$	$z_{2,n}$	(r_2, ID_2, ϱ_2)
$\mathcal{U}_3 \Rightarrow$	$z_{3,1}$	$z_{3,2}$	$\underline{z_{3,3}}$	$\cdots$	$z_{3,n}$	(r_3, ID_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, ID_n, ϱ_n)
Key	d_1	d_2	d_3	$\cdots$	d_n	(w, Q)

- Enc: For a plaintext m, an entity[3] generates the ciphertext by the following steps:
  1. Select $\rho \in \mathbb{Z}_q^*$ and compute $c_1 = g^\rho, c_2 = w^\rho, c_3 = m \oplus H_3(Q^\rho)$.
  2. Output the ciphertext $c = (c_1, c_2, c_3)$.
- Dec: To decrypt the ciphertext $c = (c_1, c_2, c_3)$, $\mathcal{U}_i$, whose group decryption key is d_i, computes

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

We note that the communication overhead of our protocol is slightly lower that that in [26] at the Agreement stage, since no certificates are required. As to the computational overhead, our protocol is slightly less efficient at the Agreement, Enc.Key.Gen and Dec.Key.Gen stages than those in [26], but our protocol offers the authentication property. At the Enc and Dec stages, the computational overheads are the same. Regarding the storage overhead, both schemes require an entity to store the system parameter list params and a group encryption key, and, if an entity is a protocol participant, it also needs to store its group decryption key and private key. In

[3]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.

both protocols, the group encryption key, group decryption key and private key are of the same size. However, the length of params in [26] is related to the largest number of participants in the group. Hence, the length of params is great if the largest number of participants is great. The length of params in our protocol is constant and small.

We next show that the security of our IB-AAGKA protocol can be reduced to the difficulty of solving the k-BDHE problem.

Theorem 2. *Suppose an adversary $\mathcal{A}$ who asks at most q_{H_1} queries to H_1, q_{H_2} queries to H_2, q_{H_3} queries to H_3, q_c queries to* $\mathsf{Corrupt}$*, q_s queries to* Send*, q_{er} queries to* $\mathsf{Ek.Reveal}$ *and q_{dr} queries to* $\mathsf{Dk.Reveal}$*, 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*

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

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. 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 number of participants in the group is at most k. As defined in Section 3.1, when a new session is first initiated, it will have a unique session ID. Denote the session ID by 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.

$\mathsf{Send}(\Pi_{\mathcal{U}_i}^\pi, \Delta)$: $\mathcal{C}$ maintains an initially empty list S^{list}. Assume $\mathsf{pid}_{\mathcal{U}_i}^\pi = \{ID_1, ..., ID_n\}$. To answer this query, $\mathcal{C}$ first recovers $c_{\mathsf{sid}_{\mathcal{U}_i}^\pi}$ corresponding to $\mathsf{sid}_{\mathcal{U}_i}^\pi$, 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 $c_{\mathsf{sid}_{\mathcal{U}_i}^\pi} = 0$ 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 $0 \le j \le n$, compute $z_{i,j} = s_i f_j^{\eta_i}$.
3. Add $(ID_i, \mathsf{sid}_{\mathcal{U}_i}^{\pi}, \eta_i, z_{i,i})$ to S^{list} and publish $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\dots,n\}, j\neq i})$.

- Else if $c_{\mathsf{sid}_{\mathcal{U}_i}^{\pi}} = 0$ 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 $1 \le j \le n$, $z_{i,j} = r_i^{\nu_j} g_1^{\mu_i} y_j^{\eta_i} \prod_{l=1}^{n, l\neq j} y_{k-l+1+j}^{-1}$.
   3. Add $(ID_i, \mathsf{sid}_{\mathcal{U}_i}^{\pi}, \eta_i, z_{i,i})$ to S^{list} and publish $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\dots,n\}, j\neq i})$.

- Else if $c_{\mathsf{sid}_{\mathcal{U}_i}^{\pi}} = 1$ and $c_{H_1^i} = 0$, perform the following steps:

   1. Select $\eta_i \in \mathbb{Z}_q^*$ at random and compute $r_i = g^{\eta_i} y_{k-i+1}$.
   2. For $1 \le j \le n, i \neq j$, compute $z_{i,j} = r_i^{\nu_j} g_1^{\mu_i} y_j^{\eta_i} y_{k-i+1+j}$.
   3. Add $(ID_i, \mathsf{sid}_{\mathcal{U}_i}^{\pi}, \eta_i, null)$ to S^{list} and respond with $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\dots,n\}, j\neq i})$.

- Else, $c_{\mathsf{sid}_{\mathcal{U}_i}^{\pi}} = 1$ and $c_{H_1^i} = 1$, do the following:

   1. Randomly select $\eta_i \in \mathbb{Z}_q^*$ and compute $r_i = g^{\eta_i} \prod_{l=1}^{n, l\neq i} y_{k-l+1}^{-1}$.
   2. For $1 \le j \le n, i \neq j$, 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 $(ID_i, \mathsf{sid}_{\mathcal{U}_i}^{\pi}, \eta_i, null)$ to S^{list} and respond with $(r_i, \varrho_i, \{z_{i,j}\}_{j\in\{1,\dots,n\}, j\neq i})$.

$\mathsf{Ek.Reveal}(\Pi_{\mathcal{U}_i}^{\pi})$: Assume $\mathsf{ms}_{\mathcal{U}_i}^{\pi} = \{\sigma_1, \dots, \sigma_n\}$, where $\sigma_l = (r_l, \varrho_l, \{z_{l,j}\}_{j\in\{1,\dots,n\}, j\neq l})$. If $\mathsf{state}_{\mathcal{U}_i}^{\pi} = \mathit{successfully\ terminated}$, $\mathcal{C}$ 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 $null$.

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

- If $c_{\mathsf{sid}_{\mathcal{U}_i}^{\pi}} = 1$, 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 $\mathsf{ms}_{\mathcal{U}_i}^{\pi} = \{\sigma_1, \dots, \sigma_n\}$, where $\sigma_l = (r_l, \varrho_l, \{z_{l,j}\}_{j\in\{1,\dots,n\}, j\neq l})$; compute and output $d_i = \prod_{l=1}^{n} z_{l,i}$.

$\mathsf{Test}(\Pi_{\mathcal{U}_i}^{\pi})$: At some point, $\mathcal{A}$ chooses a fresh $\Pi_{\mathcal{U}_i}^{\pi}$ and two messages m_0, m_1 on which it wishes to be challenged. Assume $\mathsf{pid}_{\mathcal{U}_i}^{\pi} = \{ID_1^*, \dots, ID_n^*\}$, $\mathsf{ms}_{\mathcal{U}_i}^{\pi} = \{\sigma_1^*, \dots, \sigma_n^*\}$, where $\sigma_l^* = (r_l^*, \varrho_l^*, \{z_{l,j}^*\}_{j\in\{1,\dots,n\}, j\neq l})$ 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 ($\mathsf{Event\ 3}$).
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^*, $\mathsf{ek}^{\pi}_{\mathcal{U}_i} = (w^*, Q^*)$ equals $(g^{\sum_{l=1}^n \eta_l^*}, \hat{e}(g^{\sum_{l=1}^n \mu_l^*} y_k, g_1))$.
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 he 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 any inconsistency 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 $\mathsf{Event\ 1}$ or $\mathsf{Event\ 2}$ or $\mathsf{Event\ 3}$ happens. We must calculate $\Pr[\neg\mathsf{Event\ 1} \wedge \neg\mathsf{Event\ 2} \wedge \neg\mathsf{Event\ 3}]$.

By our setting, it is easy to get $\Pr[\neg\mathsf{Event\ 1}] \geq (1-\delta)^{q_c}$, $\Pr[\neg\mathsf{Event\ 2}] \geq (1-\delta)^{q_{dr}}$, $\Pr[\neg\mathsf{Event\ 3}] \geq \delta^2(1-\delta)^{n-1}(1 - n\mathsf{Adv}_{sig}(\mathcal{A}))$. 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

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

In conclusion, we have that 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

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

$\square$

5.2. From CPA to CCA

As mentioned in Section 3.2, the most powerful attack against IB-AAGKA is Ind-ID-CCA. To achieve security against Ind-ID-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]. In this section, we show how to construct an Ind-ID-CCA secure IB-AAGKA with the Fujisaki-Okamoto conversion [16], which is used by Boneh and Franklin [6] to convert an Ind-ID-CPA secure identity-based encryption scheme into a Ind-ID-CCA secure one. The improved protocol comes as follows.

The first five algorithms/protocols Setup, Extract, Agreement, Enc.Key.Gen and Dec.Key.Gen are the same as those in Section 5.1 except that in the new Setup, two additional hash functions $H_4 : \{0,1\}^{\varsigma} \times \{0,1\}^{\varsigma} \longrightarrow \mathbb{Z}_q^*$ and $H_5 : \{0,1\}^{\varsigma} \longrightarrow \{0,1\}^{\varsigma}$ are chosen. The main differences are the Enc and Dec algorithms, which are modified as follows:

- Enc: For a plaintext m, an entity generates the ciphertext by the following steps:
  1. Select $\vartheta \in \{0,1\}^{\varsigma}$, compute $\rho = H_4(\vartheta, m)$, $c_1 = g^{\rho}, c_2 = w^{\rho}, c_3 = \vartheta \oplus H_3(Q^{\rho}), c_4 = m \oplus H_5(\vartheta)$.
  2. Output the ciphertext $c = (c_1, c_2, c_3, c_4)$.
- Dec: To decrypt the ciphertext $c = (c_1, c_2, c_3, c_4)$, $\mathcal{U}_i$, whose group decryption key is d_i:
  1. Compute $\vartheta = c_3 \oplus H_3(\hat{e}(d_i, c_1)\hat{e}(f_i^{-1}, c_2))$;
  2. Compute $m = c_4 \oplus H_5(\vartheta)$;
  3. Set $\rho = H_4(\vartheta, m)$, test $c_1 \overset{?}{=} g^{\rho}, c_2 \overset{?}{=} w^{\rho}$; if yes, output m; otherwise, reject the ciphertext.

5.3. Trade-Off Between Communication and Ciphertext Size

In the above protocols, the transmission overhead of each participant is $\mathcal{O}(n)$ group elements, for a group of size n. In the following, we introduce a method to decrease the transmission overhead to $\mathcal{O}(\sqrt{n})$ group elements.

To reduce the transmission overhead, we can separate the group into $\lceil \sqrt{n} \rceil$ subgroups, where $\lceil \sqrt{n} \rceil$ is the smallest integer no less than $\sqrt{n}$. The participants within each subgroup can run our above protocol to generate their subgroup encryption and decryption keys. By this setting, we will have $\lceil \sqrt{n} \rceil$ subgroup encryption keys and each participant will have a subgroup decryption key. Knowing all the subgroup encryption keys, a sender may separately encrypt to each subgroup under the corresponding subgroup encryption key, and generate the final ciphertext by concatenating all of the underlying individual ones. To decrypt the ciphertext, a subgroup member first extracts the subciphertext corresponding to his subgroup from the full ciphertext, and then decrypts the subciphertext using his subgroup decryption key with our Dec algorithm. By the above modification, each participant only needs to transmit $\mathcal{O}(\sqrt{n})$ group elements to enable a sender to send a secret message to n members, at the cost of a ciphertext of $\mathcal{O}(\sqrt{n})$ group elements during encryption.

6. Insider Attacks

Security of a cryptographic protocol may depend on the behavior of its participants. Obviously, it is more challenging for a protocol to guarantee security properties when legitimate participants are malicious or dishonest, and do not act according to the protocol specification. In fact, our IB-AGKA protocols enjoy the *key-compromise impersonation resilience* [18] property and may *identify malicious participants.*

Key-compromise impersonation resilience. We say a key agreement protocol is key-compromise impersonation resilient if compromising a participant A's long-term private key will allow an adversary to impersonate A, but it should not enable the adversary to impersonate entities other than A. In our IB-AAGKA protocol, if an adversary wants to impersonate an entity B other than A, the adversary must know the private key of B, since the adversary needs to use B's private key to generate a valid signature (a valid signature is impossible to forge by an attacker who does not know the private key of the signer, *i.e.*, the group member B). Therefore, it is easy to see that our protocol has the key-compromise impersonation resilience property.

Identify malicious participants. A malicious participant may just broadcast some authenticated random strings during the protocol execution to paralyze the protocol. In most of the existing GKA protocols, it is difficult to identify who is the actual malicious participant even if the protocols are authenticated and proven secure against active attackers. As mentioned in [26], this is because the authentication in those GKA protocols only prevents active attacks from outside attackers. To withstand inside attackers, one might resort to zero-knowledge proofs, but doing so would probably degrade the efficiency of the protocol in a substantial way. For our IB-AAGKA protocol, essentially, the transcripts from participants are signatures. Hence, this kind of attacks can be easily detected.

7. 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. An efficient tradeoff has been proposed to balance the size of the ciphertext and the size of the protocol transcript for each member. The resulting scheme has sublinear complexity in both communication and computation. Also, of independent interest, an efficient IB-B-MS scheme is proposed to allow multiple signers to sign multiple messages in an efficient way. The batch multi-signature can be separated into individual multi-signatures, which may be an interesting property for some applications.

Disclaimer and acknowledgments

This work is partly supported by the Spanish Government under projects TSI2007-65406-C03-01 "E-AEGIS", TIN2009-11689 "RIPUP", "eVerification" TSI-020100-2009-720, SeCloud TSI-020302-2010-153 and CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", by the Nature Science Foundation of China through projects 60970114, 60970115, 60970116 and 61003214, and by the Fundamental Research Funds for the Central Universities of China through project 3103004 and is partly Scientific Research Program Funded by Shaanxi Provincial Education Department (Program No. 2010JK727). The fourth author is partially supported as an ICREA-Acadèmia researcher by the Catalan Government. The authors are with the UNESCO Chair in Data Privacy, but this paper does not necessarily reflect the position of UNESCO nor does it commit that organization.

References

[1] M. Bellare, R. Canetti, H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange, in: Proceedings of STOC 1998, ACM, 1998, pp. 419-428.

[2] M. Bellare, P. Rogaway, Entity authentication and key distribution, in: Proceedings of CRYPTO 1993, LNCS, vol. 773, Springer-Verlag, 1994, pp. 232-249.

[3] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: Proceedings of ACM CCS 1993, ACM, 1993, pp. 62-73.

[4] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated key exchange secure against dictionary attacks, in: Proceedings of EUROCRYPT 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 139-155.

[5] D. Boneh, X. Boyen, E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, in: Proceedings of EUROCRYPT 2005, LNCS, vol. 3494, Springer-Verlag, 2005, pp. 440-456.

[6] D. Boneh, M. Franklin, Identity based encryption from the Weil pairing, SIAM Journal of Computing 32(3)(2003) 586-615.

[7] E. Bresson, O. Chevassut, D. Pointcheval, Provably authenticated group Diffie-Hellman key exchange - The dynamic case, in: Proceedings of ASIACRYPT 2001, LNCS, vol. 2248, Springer-Verlag, 2001, pp. 290-309.

[8] E. Bresson, O. Chevassut, D. Pointcheval, Dynamic group Diffie-Hellman key exchange under standard assumptions, in: Proceedings of EUROCRYPT 2002, LNCS, vol. 2332, Springer-Verlag, 2002, pp. 321-336.

[9] E. Bresson, O. Chevassut, D. Pointcheval, J. Quisquater, Provably authenticated group Diffie-Hellman key exchange, in: Proceedings of ACM CCS 2001, ACM, 2001, pp. 255-264.

[10] M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, in: Proceedings of EUROCRYPT 1994, LNCS, vol. 950, Springer-Verlag, 1995, pp. 275-286.

[11] J. Camenisch, S. Hohenberger, M. Pedersen, Batch verification of short signatures, in: Proceedings of EUROCRYPT 2007, LNCS, vol. 4515, Springer-Verlag, 2007, pp. 246-263.

[12] X. Cao, W. Kou, X. Du, A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges, Information Sciences 180(15)(2010) 2895-2903.

[13] T. Chang, M. Hwang, W. Yang, A communication-efficient three-party password authenticated key exchange protocol, Information Sciences 181(1)(2011) 217-226.

[14] K. Choi, J. Hwang, D. Lee, Efficient ID-based group key agreement with bilinear maps, in: Proceedings of PKC 2004, LNCS, vol. 2947, Springer-Verlag, 2004, pp. 130-144.

[15] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory 22(6)(1976) 644-654.

[16] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: Proceedings of CRYPTO 1999, LNCS, vol. 1666, Springer-Verlag, 1999, pp. 537-554.

[17] C. Gentry, Z. Ramzan, Identity-based aggregate signatures, in: Proceedings of PKC 2006, LNCS, vol. 3958, Springer-Verlag, 2006, pp. 257-273.

[18] M. Gorantla, C. Boyd, J. Nieto, Modeling key compromise impersonation attacks on group key exchange protocols, in: Proceedings of PKC 2009, LNCS, vol. 5443, Springer-Verlag, 2009, pp. 105-123.

[19] H. Guo, Z. Li, Y. Mu, X. Zhang, Provably secure identity-based authenticated key agreement protocols with malicious private key generators, Information Sciences 181(3)(2011) 628-647.

[20] A. Joux, A one round protocol for tripartite Diffie-Hellman, Journal of Cryptology 17(4)(2004) 263-276.

[21] J. Katz, M. Yung, Scalable protocols for authenticated group key exchange, in: Proceedings of CRYPTO 2003, LNCS, vol. 2729, Springer-Verlag, 2003, pp. 110-125.

[22] J. Katz, J. Shin, Modeling insider attacks on group key-exchange protocols, in: Proceedings of ACM CCS 2005, ACM, 2005, pp. 180-189.

[23] H. Kim, S. Lee, D. Lee, Constant-round authenticated group key exchange for dynamic groups, in: Proceedings of ASIACRYPT 2004, LNCS, vol. 3329, Springer-Verlag, 2004, pp. 245-259.

[24] C. Kudla, K. Paterson, Modular security proofs for key agreement protocols, in: Proceedings of ASIACRYPT 2005, LNCS, vol. 3788, Springer-Verlag, 2005, pp. 549-565.

[25] A. Shamir, Identity based cryptosystems and signature schemes, in: Proceedings of CRYPTO'84, LNCS vol. 196, Springer-Verlag, 1984, pp. 47-53.

[26] Q. Wu, Y. Mu, W. Susilo, B. Qin, J. Domingo-Ferrer, Asymmetric group key agreement, in: Proceedings of EUROCRYPT 2009, LNCS, vol. 5479, Springer-Verlag, 2009, pp. 153-170.

[27] L. Zhang, Q. Wu, B. Qin, J. Domingo-Ferrer, Identity-based authenticated asymmetric group key agreement protocol, in: Proceedings of COCOON 2010, LNCS, vol. 6196, Springer-Verlag, 2010, pp. 510-519.

[28] L. Zhang, B. Qin, Q. Wu, F. Zhang, Efficient many-to-one authentication with certificateless aggregate signatures, Computer Networks 54(14)(2010) 2482-2491.

[29] L. Zhang, F. Zhang, Q. Wu, J. Domingo-Ferrer, Simulatable certificateless two-party authenticated key agreement protocol, Information Sciences 180(6)(2010) 1020-1030.