Round-Efficient and Sender-Unrestricted Dynamic Group Key Agreement Protocol for Secure Group Communications

Lei Zhang, *Member, IEEE,* Qianhong Wu, *Member, IEEE,* Josep Domingo-Ferrer, *Fellow, IEEE,* Bo Qin, Zheming Dong

Abstract—Modern collaborative and group-oriented applications typically involve communications over open networks. Given the openness of today's networks, communications among group members must be secure and, at the same time, efficient. Group key agreement is widely employed for secure group communications in modern collaborative and group-oriented applications. This paper studies the problem of group key agreement in identity-based cryptosystems with an emphasis on round-efficient, sender-unrestricted, member-dynamic and provably secure key escrow freeness. The problem is resolved by proposing a one-round dynamic asymmetric group key agreement protocol which allows a group of members to dynamically establish a public group encryption key while each member has a different secret decryption key in an identity-based cryptosystem. Knowing the group encryption key, any entity can encrypt to the group members so that only the members can decrypt. We construct this protocol with a strongly unforgeable stateful identity-based batch multi-signature scheme. The proposed protocol is shown to be secure under the k-bilinear Diffie-Hellman exponent assumption.

Keywords: communication security, key management, identity-based cryptography, asymmetric group key agreement.

1 Introduction

Secure group communication is usually required in modern collaborative and distributed applications such as multi-party interactive computations, peer-to-peer file sharing and distributed social networks. A popular approach to secure group communications is to exploit group key agreement (GKA). Conventional GKA protocols allow a group of members to interact over an open network to establish a common secret key; thereafter, the group members can securely exchange messages using this shared key. This implies that, when a sender wants to send a secret message to a group of receivers, the sender has to first join the receivers to form a group and run a GKA protocol. This is inefficient since the sender may change frequently. We call this limitation of conventional GKA *sender restriction*. Further, with the standard round notion, the best-known GKA protocols require two or more rounds to establish a secret key.

Due to the above two limitations of conventional GKA protocols, they are ill-suited to scenarios like the following ones. *Scenario 1*. Two (or more) combat units would like to securely communicate with each other to coordinate their actions in different battlefields; any member of one unit may wish to report to the other unit. *Scenario 2*. A group of users in different time zones would like to discuss some sensitive topics via an untrusted third party, e.g., a social network service provider (such as Facebook). If conventional GKA is used in Scenario 1, each member of one unit has to first run the GKA protocol with the members of the other unit in a different battlefield. This is almost prohibitive given the poor communication environment in battle scenarios. In Scenario 2, if a GKA protocol with two or more rounds is used, all users have to stay online to finish the protocol before they can receive any encrypted contents. This is difficult for users in different time zones. Motivated by the above observations, Wu *et al.* [20] introduced the notion of asymmetric group key agreement (AGKA) and proposed a concrete *one-round* AGKA protocol. Unlike conventional GKA, AGKA allows the members to negotiate a common group encryption key while holding different decryption keys. Any user may access the group encryption key and securely encrypt to the group members. Thus, AGKA is *sender-unrestricted*.

The original AGKA notion and the instantiated protocol were only intended for static groups and were only secure against passive attackers who just eavesdrop the open communications. In the real world, most group communication environments are dynamic, meaning that users can join or leave a group frequently. Further, security against passive attackers is not sufficient because realistic attackers may fully control open networks and launch powerful active attacks such as member impersonation, communication tampering, replay of early protocol transcripts, etc. [14], [16]. To resist active attacks, an authenticated AGKA protocol

- *Lei Zhang and Zheming Dong are with the Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, China; Qianhong Wu is with the School of Electronic and Information Engineering, Beihang University, China; Josep Domingo-Ferrer is with the Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia; Bo Qin is with the School of Information, Renming University, China (e-mail: leizhang@sei.ecnu.edu.cn, qhwu@xidian.edu.cn, josep.domingo@urv.cat, bo.qin@ruc.edu.cn, dongzm@ecnu.cn).*

[21] for static groups has been proposed in the identity-based cryptosystem (IBC) setting [1]. In this IBC setting, a key generation center (KGC) is employed to generate the long-term private keys for the group members. With these private keys, the members can then securely establish a secure broadcast channel among them. The authenticated AGKA protocol in [21] achieves partial forward secrecy. That is, if only one or some specific group members' private keys are compromised, the secrets exchanged before the compromise stay unknown to the attacker. However, if all the group members' private keys are leaked, then the previously established secrets will be exposed to the attacker and the protocol is no longer secure. In practice, we do not know which members might be compromised after the protocol is deployed; in the worst case, all the members and even the KGC would be compromised. Obviously, since the KGC knows all the long-term private keys of the group members, it can always read the secrets. This is known as the key escrow problem. This observation motivates us to investigate authenticated AGKA protocols with stronger active security.

1.1 Our Contributions

In [24], a static identity-based authenticated asymmetric group key agreement (IBAAGKA) protocol without key escrow was proposed, which enables a group of users to establish a common encryption key and their respective decryption keys in one round. Although the protocol does not need certificates and is free from key escrow, extra efforts are required to address user dynamicity and provable security. In this paper, we concentrate on dynamic authenticated AGKA in the IBC setting. The contributions of this paper include the following aspects.

We first formalize the notion of dynamic IBAAGKA without key escrow. In this model, a trusted KGC is employed to generate the long-term private keys of group members. Then the members can establish a public group encryption key so that they can securely receive messages encrypted with this group encryption key. Further, we also allow users to leave and join the group. Our notion captures the typical active security properties of secrecy and known-key security [21], [23] derived from their analogs in conventional authenticated GKA protocols. The former means that only the group members can read the message exchanged after the AGKA protocol is executed. The latter means that an attacker who knows the group decryption keys of previous sessions cannot compute subsequent group decryption keys. Furthermore, our notion also captures *key escrow freeness* [6], which means that even if an attacker corrupts the KGC, he cannot read any secret messages exchanged before the corruption. Since an attacker compromising the KGC can compute the long-term private key of any member, this key escrow freeness implies perfect forward secrecy [2] which requires that an attacker cannot break the secrecy of previous protocol runs even if the attacker obtains all the members' long-term private keys. We argue that key escrow freeness is important because, in (dynamic) IBAAGKA protocols, the KGC is the Achilles's heel and the most vulnerable spot for an attacker to break.

Our design of a dynamic authenticated AGKA protocol is based on strongly unforgeable stateful identity-based batch multi-signatures (IBBMS) [24]. This notion allows a signer to generate a batch identity-based signature on a batch of messages carrying an embedded piece of state information. Signatures from multiple signers, provided that these signatures were generated on the same messages under the same state information, can be partially aggregated into a batch multi-signature. The individual multi-signatures are valid if and only if the resulting batch multi-signature is valid. As to the security of a strongly unforgeable stateful IBBMS scheme, it is required that an attacker cannot even produce a new multi-signature for a previously signed message under the same state information, even if the attacker is allowed to adaptively choose messages and state information to query signatures. We instantiate a strongly unforgeable stateful IBBMS scheme from bilinear maps. In this paper, we show that the scheme in [24] is provably secure under the computational Diffie-Hellman assumption.

Based on the strongly unforgeable stateful IBBMS scheme, we extend the static protocol in [24] into a dynamic IBAAGKA protocol without key escrow. In the dynamic protocol, we require a group manager to record the messages sent by other members. However, unlike a trusted dealer [9] which is assumed to distribute secret keys, the group manager could be any member of the group and even leave the group. Even if the long-term private key of the group manager is corrupted, the previously generated decryption keys of group members remain secure. The dynamic protocol is a one-round protocol. Specifically, if a member leaves the group, the group manager should only broadcast a message to other members. Further, if a user wants to join the group, the user only needs to send a message to the other members. The security of our dynamic protocol is proven under the k-bilinear Diffie-Hellman exponent assumption (which is widely used in recent cryptographic constructions) and the strong unforgeability of the stateful IBBMS. Though our protocol achieves the strongest active security among AGKA protocols so far, its computation and communication complexities are comparable to those of up-to-date AGKA protocols.

1.2 Related Work

GKA has attracted a lot of attention in cryptography. However, the best-known proposals [5], [13], [18] require two or more rounds to obtain a secret key and an additional round for each member to confirm the established secret key. Though multilinear maps [4], [11] can be used to realize one-round GKA protocols, an additional round for key confirmation is still required. Further, as discussed above, conventional GKA protocols only support secure intra-group communications. GKA protocols have also been proposed that use identity-based cryptography, which alleviates the complicated certificate management associated to the PKI setting [8], [17]. These protocols require two or more rounds and are sender-restricted.

The first one-round (A)GKA protocol was proposed by Wu *et al.* [20]. Unlike in previous GKA protocols, the

keys established by the members are, respectively, a secret group decryption key for each member and a public group encryption key. Any sender (even one not in the group) who knows the group encryption key may encrypt to the members. Each member may decrypt the corresponding ciphertext by using her own group decryption key. Since different keys are used for encryption and decryption, this kind of GKA protocols are called asymmetric GKA (AGKA) protocols. Another interesting property of Wu *et al.*'s AGKA protocol is that it allows key confirmation without additional communication. A member just needs to test whether a message m is equal to the message obtained from first encrypting m under the group encryption key and then decrypting the corresponding ciphertext using her group decryption key. If the messages are equal, the member has obtained correct keys. One may note that a trivial one-round AGKA can be constructed by concatenating several public-key encryptions. A sender just needs to first encrypt to each member separately and then generate the final ciphertext by concatenating all the underlying individual ones. However, this solution leads to $\mathcal{O}(n)$-size ciphertext and requires $\mathcal{O}(n)$ encryption operations for a group of n receivers, while the protocol in [20] only requires $\mathcal{O}(1)$-size ciphertext and $\mathcal{O}(1)$ encryption operations. The challenge is to design one-round AGKA protocols with efficient encryption and short ciphertexts.

The first authenticated AGKA protocol was proposed in [21], [23]; it is one-round and can handle sender changes efficiently. However, an additional identity-based signature is required to guarantee the security of the protocol, which reduces its interest. Further, it only achieves partial forward secrecy (PFS). To achieve key escrow freeness, it seems that we may add random secret value(s) [7] in the key agreement phase of protocol in [21], [23], as in conventional GKA protocols. However, it is unclear how to use this method without affecting the round efficiency of AGKA protocols. We note that AGKA was later extended to certificateless public key cryptosystems [15], [19], [22], which can be viewed as a variant of IBC. Nevertheless, no formal security analysis is given for the protocol in [19]. The protocol in [15] is proven in a formal security model, but the known-key security property is not modeled. The protocol in [22] is proposed with a formal security analysis. However, similar to the AGKA protocol in [21], [23], it only achieves PFS.

The rest of this paper is organized as follows. Section 2 contains background. Section 3 defines the security of dynamic IBAAGKA protocols. Section 4 is a strongly unforgeable stateful IBBMS scheme. Section 5 proposes our dynamic protocol. An evaluation of the protocol is given in Section 6. Finally, Section 7 is a conclusion.

2 Background

2.1 Problem Statement

We consider the situation of a sender who wants to securely transmit messages to a group of receivers. The problem is how can the sender do this in an environment with the following constraints:

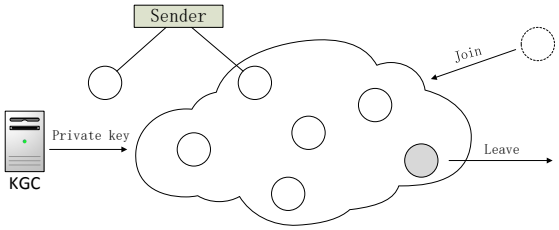


Fig. 1: Network model

1) A fully trusted dealer to generate keys for the group members is not available;
2) It is hard to estimate who will send encrypted messages to the group members;
3) The system is key escrow free;
4) The group is dynamic, that is, a user may join or leave the group.

It is worth noticing that broadcast encryption [9] may also perform a similar function to AGKA. However, in a broadcast encryption system, a trusted dealer is usually required to maintain the group. Even though some broadcast encryption systems are free from trusted dealers, they cannot offer forward secrecy and/or key escrow freeness [12].

2.2 Network Model

Fig. 1 illustrates the network model. It consists of a KGC, a sender and a group of protocol participants.

The KGC is a trusted authority. It issues private keys for the protocol participants. The protocol participants run our IBAAGKA protocol to establish a group encryption key and respective secret decryption keys for each participant. At any time, a protocol participant may leave the group. A user may also join the group if the number of the protocol participants is smaller than the maximum allowable group size. A sender can be a protocol participant or any user not in the group.

2.3 Bilinear Maps and Complexity Assumptions

Our constructions are from bilinear maps. 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: $\hat{e}(g^\alpha, g^\beta) = \hat{e}(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q^*$. 2) Non-degeneracy: There exist $u, v \in \mathbb{G}_1$ 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 constructions is based on the assumptions that the computational Diffie-Hellman (CDH) and k-bilinear Diffie-Hellman exponent (BDHE) [3] problems are hard.

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

k-BDHE Problem: Given g, h and $g_i = g^{\alpha^i}$ in $\mathbb{G}_1$ for $i = 1, 2, ..., k, k+2, ..., 2k$ as input, compute $\hat{e}(g, h)^{\alpha^{k+1}}$.

3 Security Model

3.1 Notations

Let $\mathbb{P}$ be a polynomial-size set of participants. Any subset $\mathbb{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\} \subseteq \mathbb{P}$ may launch an IBAAGKA protocol. We define the following notations which will be used in our security model:

- $\Pi^{\pi}_{\mathcal{U}_i}$ represents instance π of participant $\mathcal{U}_i$.
- $ID^{\pi}_{\mathcal{U}_i}$ is the current identity associated with $\Pi^{\pi}_{\mathcal{U}_i}$. If the protocol is static, $ID^{\pi}_{\mathcal{U}_i}$ is always equal to the real identity ID_i of $\mathcal{U}_i$. In our dynamic protocol, each participant will obtain several private keys along with different indices from the KGC. Thus, in the dynamic case, $ID^{\pi}_{\mathcal{U}_i}$ contains ID_i and the current index of $\mathcal{U}_i$'s private key.
- $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ is the *partner ID* of $\Pi^{\pi}_{\mathcal{U}_i}$. It contains the current 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.
- $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$ is the unique *session ID* of instance $\Pi^{\pi}_{\mathcal{U}_i}$. All members taking part in a given execution of a protocol have the same session ID. In our protocol, we will set the session ID to be the concatenation of $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$, a time interval (for example, one day specified as a date) and a counter of the number of sessions executed by the participants with *partner ID* $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ in the time interval.
- $\mathsf{isid}^{\pi}_{\mathcal{U}_i}$ is the initial *session ID* of instance $\Pi^{\pi}_{\mathcal{U}_i}$. In a static IBAAGKA protocol, $\mathsf{isid}^{\pi}_{\mathcal{U}_i}$ is equal to $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$. However, in the dynamic case, $\mathsf{isid}^{\pi}_{\mathcal{U}_i}$ is set to be the *session ID* of the protocol when it is first initiated.
- $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$ is the concatenation of all messages sent and received by $\Pi^{\pi}_{\mathcal{U}_i}$ during its execution, where the messages are ordered by the indices of the protocol participants.
- $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ is the group encryption key held by $\Pi^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{dk}^{\pi}_{\mathcal{U}_i}$ is the group decryption key held by $\Pi^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{state}^{\pi}_{\mathcal{U}_i}$ represents the current (internal) state of instance $\Pi^{\pi}_{\mathcal{U}_i}$. $\Pi^{\pi}_{\mathcal{U}_i}$ is *terminated*, if it stops sending; and it is *accepted* if $\Pi^{\pi}_{\mathcal{U}_i}$ is *terminated* and no incorrect behavior has been detected, i.e., it possesses $\mathsf{ek}^{\pi}_{\mathcal{U}_i}(\neq null)$, $\mathsf{dk}^{\pi}_{\mathcal{U}_i}(\neq null)$, $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$, $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$.

Definition 1 (Partnering). Two instances $\Pi^{\pi}_{\mathcal{U}_i}$ and $\Pi^{\pi'}_{\mathcal{U}_j}$ (with $i \neq j$) are partnered if and only if (1) they are accepted; (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

The security model for dynamic IBAAGKA protocols is defined by a game, which is run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In the game, $\mathcal{C}$ generates the *master-secret*, initializes the system parameters and answers various queries from $\mathcal{A}$, who controls the network communications. This game has the following stages:

Initialize: On input a security parameter ℓ, $\mathcal{C}$ generates the *master-secret* and initializes the system parameters Υ. Υ is passed to $\mathcal{A}$.

Probing: $\mathcal{A}$ may ask the following types of queries:

- $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$: It sends a message Ψ to instance $\Pi^{\pi}_{\mathcal{U}_i}$, and outputs the reply generated by this instance. In particular, if $\Psi = (\mathsf{sid}, \mathsf{pid})$, this query prompts $\mathcal{U}_i$ to initiate the protocol using session ID sid and partner ID pid. If Ψ is of incorrect format, it returns $null$.
- $\mathsf{Send}_L(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$: It sends a message Ψ to instance $\Pi^{\pi}_{\mathcal{U}_i}$ and is triggered when a participant leaves the group. In particular, if $\Psi = \perp$, this query prompts $\mathcal{U}_i$ to leave the group.
- $\mathsf{Send}_J(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$. It sends a message Ψ to instance $\Pi^{\pi}_{\mathcal{U}_i}$ and is triggered when a user joins the group. Specifically, if $\Psi = K$, it prompts $\mathcal{U}_i$ to join the group as the K-th participant and requires $\Pi^{\pi}_{\mathcal{U}_i}$ to be unused.
- $\mathsf{Corrupt}(\Pi^{\pi}_{\mathcal{U}_i})$: It outputs the current private key of $\mathcal{U}_i$ associated with $\Pi^{\pi}_{\mathcal{U}_i}$ and can be used to model forward secrecy.
- $\mathsf{Corrupt}(KGC)$: It outputs the *master-secret* of the KGC and can be used to model key escrow freeness.
- $\mathsf{Ek.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: It outputs the group encryption key $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: It outputs the group decryption key $\mathsf{dk}^{\pi}_{\mathcal{U}_i}$. It is used to model known-key security.
- $\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: $\mathcal{A}$ chooses two messages (m_0, m_1) and a fresh instance $\Pi^{\pi}_{\mathcal{U}_i}$ (see Definition 2), and sends them to $\mathcal{C}$. Then $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$, encrypts m_b under $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ to produce a ciphertext c, and returns c to $\mathcal{A}$. This query can be made only once and it is used to model secrecy.

We note that the secrecy of the dynamic IBAAGKA protocols in this paper follows the definitions in [20], [21], [23], in which secrecy is defined as the indistinguishability of a message encrypted under the negotiated group encryption key from a random message in the ciphertext space.

Guess: In this last stage, $\mathcal{A}$ returns a guess $b' \in \{0, 1\}$. If $b' = b$, $\mathcal{A}$ wins the game. $\mathcal{A}$'s advantage to win the game is defined to be $\epsilon = |2\Pr[b = b'] - 1|$.

Definition 2 (Freshness). An instance $\Pi^{\pi}_{\mathcal{U}_i}$ is fresh if none of the following happens: 1) $\Pi^{\pi}_{\mathcal{U}_i}$ has not been accepted. 2) $\mathcal{A}$ has 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 any partnered instance of $\Pi^{\pi}_{\mathcal{U}_i}$. Further, $\mathcal{A}$ has queried $\mathsf{Dk.Reveal}(\Pi^{\tilde{\pi}}_{\mathcal{U}_i})$ or $\mathsf{Dk.Reveal}(\Pi^{\tilde{\pi}'}_{\mathcal{U}_j})$, where $\Pi^{\tilde{\pi}}_{\mathcal{U}_i}$ and $\Pi^{\tilde{\pi}'}_{\mathcal{U}_j}$ have the same initial session ID as $\Pi^{\pi}_{\mathcal{U}_i}$, and $ID^{\tilde{\pi}}_{\mathcal{U}_i}, ID^{\tilde{\pi}'}_{\mathcal{U}_j} \in \mathsf{pid}^{\pi}_{\mathcal{U}_i}$. 3) Before $\Pi^{\pi}_{\mathcal{U}_i}$ is accepted, $\mathsf{Corrupt}(\mathsf{KGC})$ has been queried or some participants whose current identities are in $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ have been made the Corrupt query.

Definition 3. An IBAAGKA protocol is said to be semantically indistinguishable against chosen identity and plaintext attacks (Ind-ID-CPA) if ϵ is negligible for any probabilistic polynomial-time (PPT) active adversary in the above model.

Similar to [20], [21], [23], we only define chosen-plaintext attacks (CPA) of dynamic IBAAGKA protocols. A stronger definition is security against chosen-ciphertext attacks (CCA). To achieve CCA security, some generic constructions have been proposed to convert a CPA secure encryp-

tion scheme into a CCA secure one, such as the Fujisaki-Okamoto conversion [10]. Hence, in this paper we will focus on CPA security of dynamic IBAAGKA protocols.

4 Building Block: IBBMS

In this section we propose a strongly unforgeable stateful IBBMS scheme as a building block of our IBAAGKA protocol. The latter protocol is our final goal, rather than the IBBMS scheme; yet, we will use some design principles of the IBBMS scheme to build IBAAGKA.

4.1 Definition

A stateful IBBMS scheme allows multiple signers to sign multiple messages under a piece of state information, so as to generate in an efficient way a batch multi-signature. Later, the single batch multi-signature can be separated into individual multi-signatures. The state information can be instanced by concatenating the identities of all the signers, a time interval and a counter of the number of signatures issued by these signers in the time interval. A stateful IBBMS scheme has the following five algorithms:

- BM.Setup: On input a security parameter ℓ, it generates a *master-secret* and a list of system parameters. For brevity, the system parameters are omitted as part of the inputs for the rest of algorithms.
- BM.Extract: On inputs an entity's identity ID_i and the *master-secret*, it outputs the private key of the entity.
- Sign: It takes as inputs a piece of state information $info$, t messages, a signer's identity ID_i and private key, and it outputs a batch signature on t messages.
- Aggregate: It takes as input a collection of x batch signatures by x signers on the same t messages under the same state information $info$, and it outputs a batch multi-signature.
- BM.Verify: It takes as input a batch multi-signature on t messages generated by x signers, under the same state information $info$, and it outputs either "all valid" if the batch multi-signature is valid, or a set which contains the indices of the valid multi-signatures on the corresponding messages.

4.2 The Security Model

In this section, we define the strong unforgeability of stateful IBBMS schemes. Roughly speaking, a stateful IBBMS scheme is strongly unforgeable if an adversary cannot output a different multi-signature on a message m under any state information and x signers' identities even if he can obtain the signature(s) on m under the same state information and identities. The formal definition uses the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Initialize: $\mathcal{C}$ runs BM.Setup to generate a *master-secret* and the system parameter list Υ. $\mathcal{C}$ passes Υ to $\mathcal{A}$ while keeping *master-secret* undisclosed.

Probing: $\mathcal{A}$ can adaptively issue the following queries:

- BM.Extract: The input of this query is an identity ID_i of an entity. On receiving such a query, $\mathcal{C}$ outputs the private key corresponding to ID_i.
- Sign: $\mathcal{A}$ may request a batch signature on messages $\{m_1, \ldots, m_{t_i}\}$ under an identity ID_i and a piece of state information $info_i$. On input $(ID_i, info_i, m_1, \ldots, m_{t_i})$, $\mathcal{C}$ generates a valid batch signature. If $\mathcal{A}$ requests a batch signature with a previously used state information but a different message set as input, $\mathcal{C}$ returns $null$.

Note that if $\mathcal{A}$ wants to obtain an IBBMS on messages $(m_1, \ldots, m_{t_i})$ under identities $(ID_1, \ldots, ID_x)$ and a piece of state information, he just needs to repeatedly ask Sign queries for the different identities and then generate the IBBMS by using the Aggregate algorithm.

Forgery: Eventually, $\mathcal{A}$ outputs a multi-signature σ^* on a message m^* under x identities $(ID_1^*, \ldots, ID_x^*)$ and a piece of state information $info^*$. $\mathcal{A}$ wins the above game if the following conditions are satisfied:

1) $ID_i^* \in \{ID_1^*, \ldots, ID_x^*\}$ has never been submitted to BM.Extract.
2) σ^* is not obtained by using the batch signatures output by submitting $(ID_i^*, info^*, m_1, \ldots, m_{\mathcal{I}}, \ldots, m_t)$ to the Sign queries, for $ID_i^* \in \{ID_1^*, \ldots, ID_x^*\}$, where $m_{\mathcal{I}} = m^*$ and $\mathcal{I}$ defines the index of the message.

In the last stage, $\mathcal{A}$ is not required to output a batch multi-signature, but only a single individual multi-signature. This is because a batch multi-signature can be separated into t individual multi-signatures. We only require that one of the multi-signatures be a forgery. It is also worth noticing that we require that $ID_i^* \in \{ID_1^*, \ldots, ID_x^*\}$ has never been submitted to BM.Extract. However, in a standard multi-signature scheme, an adversary usually allows querying $x - 1$ private keys corresponding to the identities in $\{ID_1^*, \ldots, ID_x^*\}$. Our restriction is stronger than the restriction in the security models for standard multi-signature schemes. We note that our final goal is to build a secure dynamic IBAAGKA, not an identity-based batch multi-signature scheme secure in the strongest sense. Otherwise said, the positive news is that a secure dynamic IBAAGKA does not need a multi-signature scheme secure in the stronger model (which is difficult to achieve in practice): a multi-signature scheme secure in a weaker model is enough. Indeed, we will formally reduce the security of our IBAAGKA to that of our IBBMS proven in the weaker model.

Definition 4. A stateful IBBMS scheme is strongly existentially unforgeable under adaptively chosen-message attacks if and only if the success probability ϵ' of any PPT adversary in the above game is negligible.

4.3 The Scheme

- BM.Setup: On input a security parameter ℓ, KGC chooses $q, g, \mathbb{G}_1, \mathbb{G}_2, \hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ as defined in Section 2.3; chooses $\kappa \in \mathbb{Z}_q^*$ as the *master-secret* and

sets $g_{pub} = g^{\kappa}$; chooses hash functions $H_1, H_2, H_3 : \{0,1\}^* \longrightarrow \mathbb{G}_1$, $H_4 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$. The system parameter list is $\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{pub}, H_1 \sim H_4)$.

- BM.Extract: On input κ and an entity's identity ID_i, the KGC outputs $(s_{i,0} = id_{i,0}^{\kappa}, s_{i,1} = id_{i,1}^{\kappa})$, where $id_{i,0} = H_1(ID_i, 0), id_{i,1} = H_1(ID_i, 1)$.
- Sign: Let the messages to be signed be $(m_1, \ldots, m_t)$ and the state information be $info$. A signer whose identity is ID_i and private key is $(s_{i,0}, s_{i,1})$ performs the following steps:
  1) Select $\eta_i, \theta_i \in \mathbb{Z}_q^*$ and compute $r_i = g^{\eta_i}, u_i = g^{\theta_i}, v = H_2(info), \varpi_i = H_4(info, ID_i, r_i, u_i)$.
  2) Compute $f_j = H_3(info, m_j)$, $z_{i,j} = s_{i,0} s_{i,1}^{\varpi_i} v^{\theta_i} f_j^{\eta_i}$, for $1 \le j \le t$.
  3) Output the batch signature $\sigma_i = (r_i, u_i, z_{i,1}, \ldots, z_{i,t})$.
- Aggregate: Let $\{\sigma_i = (r_i, u_i, z_{i,1}, \ldots, z_{i,t})\}_{1 \le i \le x}$ be a collection of signatures on the same messages $\{m_j\}_{1 \le j \le t}$ under the same $info$. Anyone can perform this algorithm to aggregate these signatures into a single batch multi-signature. Specifically, the signatures can be aggregated into $(r_1, \ldots, r_x, u_1, \ldots, u_x, d_1, \ldots, d_t)$, where $d_j = \prod_{i=1}^{x} z_{i,j}$.
- BM.Verify: To check whether the above batch multi-signature $(r_1, \ldots, r_x, u_1, \ldots, u_x, d_1, \ldots, d_t)$ is valid, a verifier computes $w = \prod_{i=1}^{x} r_i, y = \prod_{i=1}^{x} u_i, v = H_2(info)$, $f_j = H_3(info, m_j), \Gamma_j = \hat{e}(f_j, w)$ for $1 \le j \le t$, $\varpi_i = H_4(info, ID_i, r_i, u_i)$ for $1 \le i \le x$ and

$$\Omega = \hat{e}(\prod_{i=1}^{x} H_1(ID_i, 0) H_1(ID_i, 1)^{\varpi_i}, g_{pub}) \hat{e}(v, y).$$

  For $1 \le j \le t$, the verifier checks $\hat{e}(d_j, g) \stackrel{?}{=} \Omega \Gamma_j$. If all t equations hold, the verifier outputs "all valid"; else, outputs an index set $\mathbb{I}$, which means that the multi-signatures with indices in that set are valid.

The above IBBMS scheme achieves partial aggregation. For a single entity, the signature size is $t+2$ group elements. For x entities, the signature size is $x(t+2)$ group elements in total, and the IBBMS size is $2x + t$. In our dynamic IBAAGKA, we choose $t = x$. In this case, the IBBMS size is only $3/(x+2)$ of the size of the total signatures. The reduction of the signature size after aggregation is significant.

Identity-based signature schemes are usually randomized. To construct an IBBMS scheme supporting full aggregation, one has to find a way to "aggregate the randomness" provided by the multiple signers; otherwise, this randomness would cause the size of the signature to grow linearly with the number of the signers. However, if the randomness can be aggregated, the IBBMS scheme is not strongly unforgeable according to the definition in Section 4.2. For instance, assume the randomness is represented by r_1 and r_2, and the aggregated randomness is $w = r_1 + r_2$. According to the definition (see Section 4.2, Forgery, condition 2), the signature generated by using as randomness $r_1 + 1$ and $r_2 - 1$ is a forgery. Although verification would not be passed before aggregation, the resulting aggregate signature is valid.

4.4 Security

The security of our scheme is based on the hardness of the CDH problem. The following claim proven in the Appendix relates the security of the IBBMS primitive to the difficulty of solving the CDH problem.

Theorem 1. If an adversary $\mathcal{A}$ can win the game in Section 4.2 with advantage ϵ' in time τ' after making at most q_{H_i} queries to H_i for $1 \le i \le 3$, q_E Extract queries, and q_σ Sign queries with maximal message size N, then there exists an algorithm to solve the CDH problem with advantage $(\frac{x+2}{q_E + q_{H_3} + x + 1})^{x+2} \frac{q_{H_3}}{e^{x+2}} \epsilon'$ in time $\mathcal{O}(\tau')$, where x is the maximum number of users in a forged IBBMS and e is Euler's number.

5 The Dynamic IBAAGKA Protocol

In this section, we propose our one-round dynamic IBAAGKA protocol. In our protocol, we need a group manager to maintain the group. We note that the group manager could be any of the protocol participants and may also leave the group. This is different from a trusted dealer. However, for better performance, we require the group manager to be relatively static.

5.1 The Protocol

The concrete protocol comes as follows:

- Setup: The same as the BM.Setup in Section 4.3, except that an additional cryptographic hash function $H_5 : \mathbb{G}_2 \longrightarrow \{0,1\}^{l_0}$ is chosen, where l_0 defines the bit-length of plaintexts. The system's parameter list is $\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{pub}, H_1 \sim H_5, l_0)$.
- Extract: Each entity may request at most N private keys. Suppose the identity of an entity is ID_i. The KGC computes $id_{i,j,0} = H_1(ID_i, j, 0), id_{i,j,1} = H_1(ID_i, j, 1)$ for $1 \le j \le N$, and outputs N private keys $\{s_{i,j,0} = id_{i,j,0}^{\kappa}, s_{i,j,1} = id_{i,j,1}^{\kappa}\}_{j \in \{1,\ldots,N\}}$. In our protocol, when a user joins the group with the same isid, he has to use a new private key. Generally, a user will not join and leave the group with the same isid frequently. Therefore, N does not need to be large in most cases.
- Agreement: Assume the group size is n and the group manager is the t-th participant in the group. This protocol runs as follows:
  1) Choose $\eta_i, \theta_i \in \mathbb{Z}_q^*$, compute $r_i = g^{\eta_i}, u_i = g^{\theta_i}$.
  2) Compute $v = H_2(\mathsf{isid}), \varpi_i = H_4(\mathsf{isid}, ID_i, \iota_i, r_i, u_i)$, where ι_i is initially set to be 1.
  3) For $1 \le j \le n$, compute $f_j = H_3(\mathsf{isid}, j)$.
  4) For $1 \le j \le n$, compute $z_{i,j} = s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_i} v^{\theta_i} f_j^{\eta_i}$.
  5) Publish $\sigma_i = (ID_i, \iota_i, r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,n\}, j \ne i})$.

TABLE 1: Messages required by the i-th user in the group

$z_{1,i}$	ID_1, ι_1, r_1, u_1
$z_{2,i}$	ID_2, ι_2, r_2, u_2
$z_{3,i}$	ID_3, ι_3, r_3, u_3
$\vdots$	$\vdots$
$z_{n,i}$	ID_n, ι_n, r_n, u_n

TABLE 2: Messages received by the group manager

$\emptyset$	$z_{1,2}$	$z_{1,3}$	$\cdots$	$z_{1,n}$	ID_1, ι_1, r_1, u_1
$z_{2,1}$	$\emptyset$	$z_{2,3}$	$\cdots$	$z_{2,n}$	ID_2, ι_2, r_2, u_2
$z_{3,1}$	$z_{3,2}$	$\emptyset$	$\cdots$	$z_{3,n}$	ID_3, ι_3, r_3, u_3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$z_{n,1}$	$z_{n,2}$	$z_{n,3}$	$\cdots$	$\emptyset$	ID_n, ι_n, r_n, u_n

Each participant $\mathcal{U}_i, i \neq t$ maintains a table like Table 1. If $\mathcal{U}_i$ is the group manager, he has to maintain a table like Table 2. $\iota_1, ..., \iota_n$ are initially set to 1. We note that for the i-th participant, all the $z_{i,j}$ are published, except $z_{i,i}$. Therefore, only the i-th participant can compute d_i, which guarantees the security of the protocol.

- Leave: Assume the group manager is the t-th participant and the l-th participant is leaving the group.
  - If $l \neq t$, do the following:
    1) Choose $\hat{\eta}_l, \hat{\theta}_l \in \mathbb{Z}_q^*$, compute $\hat{r}_l = g^{\hat{\eta}_l}, \hat{u}_l = g^{\hat{\theta}_l}$, set $\iota_t = \iota_t + 1$, and compute $\hat{\varpi}_l = H_4(\mathsf{isid}, ID_t, \iota_t, \hat{r}_l, \hat{u}_l)$.
    2) For $1 \leq j \leq n$, compute $\hat{z}_{l,j} = s_{t,\iota_t,0} s_{t,\iota_t,1}^{\hat{\varpi}_l} v^{\hat{\theta}_l} f_j^{\hat{\eta}_l}$, where $v = H_2(\mathsf{isid})$, $f_j = H_3(\mathsf{isid}, j)$.
    3) Publish $(ID_t, \iota_t, \hat{r}_l, \hat{u}_l, \{\hat{z}_{l,j}\}_{j \in \{1,...,n\}, j \neq l})$.

    All the participants have to update the elements in the l-th row of the tables maintained by themselves. Specifically, if the participant is the group manager $\mathcal{U}_t$, he updates $z_{l,j} = \hat{z}_{l,j}, j \in \{1, \ldots, n\} \backslash l$, $ID_l = ID_t, r_l = \hat{r}_l, u_l = \hat{u}_l$ in Table 2; else, the i-th participant updates $z_{l,i} = \hat{z}_{l,i}, ID_l = ID_t, \iota_l = \iota_t, r_l = \hat{r}_l, u_l = \hat{u}_l$ in the table maintained by himself.
  - Else, a new group manager $\mathcal{U}_{t'}$ is elected. The table maintained by $\mathcal{U}_t$ is passed to $\mathcal{U}_{t'}$. For each index i such that the i-th line of Table 2 contains identity ID_t, publish $(ID_{t'}, \iota_{t'}, \hat{r}_{t'}, \hat{u}_{t'}, \{\hat{z}_{t',j}\}_{j \in \{1,...,n\}, j \neq i})$, which is generated in the same way described above.
- Join: Assume an entity $\mathcal{U}_I$ wants to join the group as the l-th group participant.
  1) $\mathcal{U}_I$ performs the following steps:
     a) Choose $\eta_I, \theta_I \in \mathbb{Z}_q^*$, compute $r_I = g^{\eta_I}, u_I = g^{\theta_I}$, compute $v = H_2(\mathsf{isid}), \varpi_I = H_4(\mathsf{isid}, ID_I, \iota_I, r_I, u_I)$.
     b) For $1 \leq j \leq n$, compute $f_j = H_3(\mathsf{isid}, j)$ and $z_{I,j} = s_{I,\iota_I,0} s_{I,\iota_I,1}^{\varpi_I} v^{\theta_I} f_j^{\eta_I}$, set $\iota_I = \iota_I + 1$.
     c) Publish $(ID_I, \iota_I, r_I, u_I, \{z_{I,j}\}_{j \in \{1,...,n\}, j \neq l})$.
  2) The group manager updates the elements of l-th row of the Table 2 to be $z_{l,j} = z_{I,j}$ for $1 \leq j \leq n, j \neq l$, $ID_l = ID_I, \iota_i = \iota_I, r_l = r_I, u_l = u_I$. The group manager also sends $z_{i,l}, ID_i, \iota_i, r_i, u_i$ to $\mathcal{U}_I$ for $1 \leq i \leq n, i \neq l$. The i-th participant sets the l-th row of Table 1 to be $z_{l,i} = z_{I,i}, ID_l = ID_I, \iota_l = \iota_I, r_l = r_I, u_l = u_I$.
- Enc.Key.Gen: To derive the group encryption key, an entity calculates $id_{i,\iota_i,0} = H_1(ID_i, \iota_i, 0), id_{i,\iota_i,1} = H_1(ID_i, \iota_i, 1), v = H_2(\mathsf{isid}), f_j = H_3(\mathsf{isid}, j), \varpi_i = H_4(\mathsf{isid}, ID_i, \iota_i, r_i, u_i)$ for $j \in \{1, 2\}, i \in \{1, \ldots, n\}$. Define $\Delta = 1$ if Equations (1) and (2) hold, and $\Delta = 0$ otherwise.

$$\hat{e}(z_{1,2}, g) \stackrel{?}{=} \hat{e}(id_{1,\iota_1,0} id_{1,\iota_1,1}^{\varpi_1}, g_{pub}) \hat{e}(v, u_1) \hat{e}(f_2, r_1) \quad (1)$$

$$\hat{e}(\prod_{i=2}^{n} z_{i,1}, g) \stackrel{?}{=} \hat{e}(\prod_{i=2}^{n} id_{i,\iota_i,0} id_{i,\iota_i,1}^{\varpi_i}, g_{pub}) \hat{e}(v, \prod_{i=2}^{n} u_i) \times \hat{e}(f_1, \prod_{i=2}^{n} r_i) \quad (2)$$

We note that the message σ_i published by a protocol participant is essentially an IBBMS. Equations (1) and (2) are the verification algorithms of the IBBMS scheme. Equation (1) is used to guarantee that σ_1 is unmodified. Similarly, Equation (2) is used to prove that $\sigma_2, ..., \sigma_n$ are received without modification. If $\Delta = 0$, the entity aborts, because this means that some of the messages have been altered by an attacker; otherwise ($\Delta = 1$), the entity outputs the group encryption key (w, Ω), where

$$w = \prod_{i=1}^{n} r_i, \Omega = \hat{e}(\prod_{i=1}^{n} id_{i,\iota_i,0} id_{i,\iota_i,1}^{\varpi_i}, g_{pub}) \hat{e}(v, \prod_{i=1}^{n} u_i).$$

Note that a protocol participant does not need to test the value of Δ, since a similar check will be performed in the following Dec.Key.Gen stage.

- Dec.Key.Gen: Each participant $\mathcal{U}_i$ computes $w = \prod_{l=1}^{n} r_l, \Gamma_i = \hat{e}(f_i, w), d_i = \prod_{l=1}^{n} z_{l,i}$ and checks $\hat{e}(d_i, g) \stackrel{?}{=} \Omega \cdot \Gamma_i$. If the equation holds, $\mathcal{U}_i$ accepts d_i as the group decryption key and (w, Ω) generated in Enc.Key.Gen as the group encryption key; otherwise, $\mathcal{U}_i$ aborts.
- Enc: To encrypt a plaintext m, an entity selects $\rho \in \mathbb{Z}_q^*$ and computes the ciphertext $c = (c_1, c_2, c_3)$, where $c_1 = g^\rho, c_2 = w^\rho, c_3 = m \oplus H_5(\Omega^\rho)$.
- Dec: To decrypt $c = (c_1, c_2, c_3)$, $\mathcal{U}_i$ computes $m = c_3 \oplus H_5(\hat{e}(d_i, c_1) \hat{e}(f_i^{-1}, c_2))$, where d_i is the group decryption key of $\mathcal{U}_i$.

5.2 Security

Theorem 2. Let H_2, H_3 and H_5 be random oracles. Suppose that an adversary $\mathcal{A}$ makes at most q_{H_i} queries to H_i, $i \in \{2, 3, 5\}$, q_C Corrupt queries, q_S Send queries, q_{EK} Ek.Reveal queries and q_{DK} Dk.Reveal queries, and wins the game with advantage ϵ in time τ. Then there exists an

algorithm to solve the k-BDHE problem with advantage at least $\frac{n^n(1-2\epsilon')}{((q_{DK}+1)e)^n}\epsilon$ in time $T = \mathcal{O}(\tau)$ where ϵ' is the advantage of $\mathcal{A}$ in forging a valid IBBMS in time T and e is Euler's number.

Proof. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversary who can break the proposed protocol. Assume that, in each session, the group of participants is of size at most k. $\mathcal{C}$ is given $(g, h, g_1, \ldots, g_k, g_{k+2}, \ldots, g_{2k})$ of the k-BDHE problem, where $g_i = g^{\alpha^i}, i \in \{1, \ldots, k, k+2, \ldots, 2k\}$. We show how $\mathcal{C}$ can use $\mathcal{A}$ to solve the k-BDHE problem.

Initialize: When the game begins, $\mathcal{C}$ selects $\kappa \in \mathbb{Z}_q^*$ as the *master-secret* at random, and sets the system parameters $\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{pub}, H_1 \sim H_5, \zeta)$, where $g_{pub} = g^\kappa$. Υ is passed to $\mathcal{A}$.

Probing: When $\Pi^\pi_{\mathcal{U}_i}$ is initiated, $\mathcal{C}$ will flip a coin $coin^\pi_{\mathcal{U}_i}$ so that $\Pr[coin^\pi_{\mathcal{U}_i} = 1] = \delta, \quad \Pr[coin^\pi_{\mathcal{U}_i} = 0] = 1 - \delta$. The tuple $(\Pi^\pi_{\mathcal{U}_i}, coin^\pi_{\mathcal{U}_i})$ is recorded by $\mathcal{C}$. $\mathcal{C}$ answers $\mathcal{A}$'s queries as follows:

H_2 queries: $\mathcal{C}$ keeps an initially empty list H_2^{list}. On input isid_i, if there is a tuple $(\mathsf{isid}_i, \gamma_i, v_i)$ in H_2^{list}, $\mathcal{C}$ returns v_i as the answer; else, $\mathcal{C}$ randomly selects $\gamma_i \in \mathbb{Z}_q^*$, sets $v_i = g_1 g^{\gamma_i}$, adds $(\mathsf{isid}_i, \gamma_i, v_i)$ to H_2^{list} and returns v_i.

H_3 queries: $\mathcal{C}$ keeps an initially empty list H_3^{list}. On input (isid_i, j), $\mathcal{C}$ does the following:

- If there is a tuple $(\mathsf{isid}_i, j, \zeta_j, f_j)$ in H_3^{list}, return f_j.
- Else if $j \leq k$, randomly select $\zeta_j \in \mathbb{Z}_q^*$, set $f_j = g_j g^{\zeta_j}$, add $(\mathsf{isid}_i, j, \zeta_j, f_j)$ to H_3^{list} and return f_j.
- Else, randomly select $\zeta_j \in \mathbb{Z}_q^*$, set $f_j = g^{\zeta_j}$, add $(\mathsf{isid}_i, j, \zeta_j, f_j)$ to H_3^{list} and return f_j.

H_5 queries: $\mathcal{C}$ maintains an initially empty list H_5^{list}. On input a message Ψ_i, if there is a tuple (Ψ_i, ϱ_i) in H_5^{list}, $\mathcal{C}$ returns ϱ_i as the answer; otherwise, $\mathcal{C}$ randomly selects $\varrho_i \in \{0,1\}^{l_0}$, adds (Ψ_i, ϱ_i) to H_5^{list} and responds with ϱ_i.

$\mathsf{Corrupt}(\Pi^\pi_{\mathcal{U}_i})$: Suppose $ID^\pi_{\mathcal{U}_i} = (ID_i, \iota_i)$. On receiving the corrupt query, $\mathcal{C}$ outputs $(H_1(ID_i, \iota_i, 0)^\kappa, H_1(ID_i, \iota_i, 1)^\kappa)$.

$\mathsf{Corrupt}(\mathsf{KGC})$: On receiving this query, $\mathcal{C}$ outputs κ.

$\mathsf{Send}(\Pi^\pi_{\mathcal{U}_i}, \Psi)$: $\mathcal{C}$ maintains an initially empty list S^{list}. Assume $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{ID_1, \iota_1; \ldots; ID_n, \iota_n\}$, $ID^\pi_{\mathcal{U}_i} = (ID_i, \iota_i)$ and the initial session ID is isid. To answer this query, $\mathcal{C}$ first recovers $coin^\pi_{\mathcal{U}_i}$, submits isid to H_2 and (isid, j) to H_3 for $1 \leq j \leq n$, if these queries have never been issued before; then recovers $(\mathsf{isid}, \gamma, v)$ from H_2^{list} and $(\mathsf{isid}, j, \zeta_j, f_j)$ from H_3^{list} for $1 \leq j \leq n$, and computes $s_{i,\iota_i,0} = H_1(ID_i, \iota_i, 0)^\kappa, s_{i,\iota_i,1} = H_1(ID_i, \iota_i, 1)^\kappa$; and then does the following:

- If $coin^\pi_{\mathcal{U}_i} = 0$, perform the following steps:
  1) Choose $\eta_i, \theta_i \in \mathbb{Z}_q^*$, compute $r_i = g^{\eta_i}, u_i = g^{\theta_i}$.
  2) Compute $\varpi_i = H_4(\mathsf{isid}, ID_i, r_i, u_i)$.
  3) For $1 \leq j \leq n$, compute $z_{i,j} = s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_i} v^{\theta_i} f_j^{\eta_i}$.
  4) Add $(ID_i, \iota_i, \mathsf{isid}, \eta_i, \theta_i, z_{i,i}))$ to S^{list}.
  5) Output $(ID_i, \iota_i, r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,n\}, j \neq i})$.
- Else if $coin^\pi_{\mathcal{U}_i} = 1$, randomly choose $\eta_i, \theta_i \in \mathbb{Z}_q^*$ and perform the following:
  - If $i \neq n$, compute $r_i = g^{\eta_i} g_{k-i+1}, u_i = g^{\theta_i}$, $\varpi_i = H_4(\mathsf{isid}, ID_i, \iota_i, r_i, u_i)$; and for $1 \leq j \leq n, j \neq i$, compute $z_{i,j} = r_i^{\zeta_j} u_i^\gamma g_1^{\theta_i} s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_i} g_j^{\eta_i} g_{k-i+1+j}$; set $z_{i,i} = \emptyset$; add $(ID_i, \iota_i, \mathsf{isid}, \eta_i, \theta_i, z_{i,i})$ to S^{list} and output $(ID_i, \iota_i, r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,n\}, j \neq i})$.
  - Else, compute $r_i = g^{\eta_i} \prod_{l=1}^{n, l \neq i} g_{k-l+1}^{-1}, u_i = g^{\theta_i} g_k$, $\varpi_i = H_4(\mathsf{isid}, ID_i, \iota_i, r_i, u_i)$; for $1 \leq j \leq n, j \neq i$, compute
    $$z_{i,j} = r_i^{\zeta_j} u_i^\gamma g_1^{\theta_i} s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_i} g_j^{\eta_i} \prod_{l=1}^{n, l \notin \{i,j\}} g_{k-l+1+j}^{-1};$$
    output $(ID_i, \iota_i, r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,n\}, j \neq i})$ and add $(ID_i, \iota_i, \mathsf{isid}, \eta_i, \theta_i, z_{i,i})$ to S^{list}, where $z_{i,i} = \emptyset$.

$\mathsf{Send}_L(\Pi^\pi_{\mathcal{U}_i}, \Psi)$: Assume the initial session ID is isid and the K-th user $\mathcal{U}_K$ is leaving the group. If $\Psi = \perp$, $\mathcal{C}$ outputs the index of $\mathcal{U}_i$; further, if $\mathcal{U}_i$ is the group manager, a new group manager is chosen and $\mathsf{ms}^\pi_{\mathcal{U}_i}$ is also output. Else if $\mathcal{U}_i$ is not the group manager and Ψ is not from the current group manager, $\mathcal{C}$ does nothing. Else if $\mathcal{U}_i$ is not the group manager, $\mathcal{C}$ first initiates a new instance of $\mathcal{U}_i$ denoted as $\Pi^{\pi'}_{\mathcal{U}_i}$; then sets $coin^{\pi'}_{\mathcal{U}_i} = coin^\pi_{\mathcal{U}_i}$, sets $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^\pi_{\mathcal{U}_i}$ and the identity in $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ which is equal to ID_K to be $\emptyset$, sets $\mathsf{ms}^{\pi'}_{\mathcal{U}_i} = \mathsf{ms}^\pi_{\mathcal{U}_i}$, replaces the tuple(s) generated by $\mathcal{U}_K$ in $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ according to Ψ. Else $\mathcal{U}_i$ is the group manager, a new instance of $\mathcal{U}_i$ $\Pi^{\pi'}_{\mathcal{U}_i}$ is initiated. $\mathcal{C}$ sets $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ to be elements received from the old group manager if $\mathcal{U}_i$ is the newly chosen group manager or $\mathsf{ms}^\pi_{\mathcal{U}_i}$ otherwise; submits isid to H_2 and (isid, j) to H_3 for $1 \leq j \leq n$ and recovers $(\mathsf{isid}, \gamma, v)$ from H_2^{list} and $(\mathsf{isid}, j, \zeta_j, f_j)$ from H_3^{list} for $1 \leq j \leq n$; sets $\iota_i = \iota_i + 1$, computes $s_{i,\iota_i,0} = H_1(ID_i, \iota_i, 0)^\kappa, s_{i,\iota_i,1} = H_1(ID_i, \iota_i, 1)^\kappa$ and answers the query as follows:

- If $coin^{\pi'}_{\mathcal{U}_i} = 0$, do the following:
  1) Choose $\eta_K, \theta_K \in \mathbb{Z}_q^*$, compute $r_K = g^{\eta_K}, u_K = g^{\theta_K}$, $\varpi_K = H_4(\mathsf{isid}, ID_i, \iota_i, r_K, u_K)$.
  2) For $1 \leq j \leq n$, compute $z_{K,j} = s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_K} v^{\theta_K} f_j^{\eta_K}$.
  3) Add $(ID_i, \iota_i, \mathsf{isid}, \eta_K, \theta_K, z_{K,K})$ to S^{list}, set the K-th tuple in $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ to be $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j \in \{1,\ldots,n\}, j \neq K})$, set $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^\pi_{\mathcal{U}_i}$ and the K-th current identity in $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ to be $\emptyset$.
  4) Output $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j \in \{1,\ldots,n\}, j \neq K})$.
- Else if $coin^{\pi'}_{\mathcal{U}_i} = 1$ and $K \neq n$, do the following:
  1) Choose $\eta_K, \theta_K \in \mathbb{Z}_q^*$, compute $r_K = g^{\eta_K} g_{k-K+1}, u_K = g^{\theta_K}$, $\varpi_K = H_4(\mathsf{isid}, ID_i, \iota_i, r_K, u_K)$.
  2) For $1 \leq j \leq n, j \neq K$, compute $z_{K,j} = r_K^{\zeta_j} u_K^\gamma g_1^{\theta_K} s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_K} g_j^{\eta_K} g_{k-K+1+j}$.
  3) Set $z_{K,K} = \emptyset$, add $(ID_i, \iota_i, \mathsf{isid}, \eta_K, \theta_K, z_{K,K})$ to S^{list}, set the K-th tuple in $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ to

be $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j\in\{1,\ldots,n\},j\neq K})$, set $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and the K-th current identity in $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ to be $\emptyset$.

4) Output $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j\in\{1,\ldots,n\},j\neq K})$.

- Else, do the following:

1) Choose $\eta_K, \theta_K \in \mathbb{Z}^*_q$, compute $r_K = g^{\eta_K}\prod_{l=1}^{n,l\neq K} g^{-1}_{k-l+1}, u_K = g^{\theta_K} g_k$, $\varpi_K = H_4(\mathsf{isid}, ID_i, \iota_i, r_K, u_K)$.
2) For $1 \leq j \leq n, j \neq K$, compute $z_{K,j} = r_K^{\zeta_j} u_K^{\gamma} g_1^{\theta_K} s_{i,\iota_i,0} s_{i,\iota_i,1}^{\varpi_K} g_j^{\eta_K} \prod_{l=1}^{n,l\notin\{K,j\}} g^{-1}_{k-l+1+j}$.
3) Set $z_{K,K} = \emptyset$, add $(ID_i, \iota_i, \mathsf{isid}, \eta_K, \theta_K, z_{K,K})$ to S^{list}, set $\mathsf{ms}^{\pi'}_{\mathcal{U}_i} = \mathsf{ms}^{\pi}_{\mathcal{U}_i}$, set the K-th tuple in $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$ to be $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j\in\{1,\ldots,n\},j\neq K})$, set $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and the K-th current identity in $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ to be $\emptyset$.
4) Output $(ID_i, \iota_i, r_K, , u_K, \{z_{K,j}\}_{j\in\{1,\ldots,n\},j\neq K})$.

We note that, if the leaving participant was a group manager, for each index K' such that $\{ID_{K'} = \emptyset, ID_{K'} \in \mathsf{pid}^{\pi}_{\mathcal{U}_i}\}$, the new group manager also needs to publish $(ID_i, \iota_i, r_{K'}, u_{K'}, \{z_{K',j}\}_{j\in\{1,\ldots,n\},j\neq K'})$, which is generated in the same way as described above. $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ and $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ should be also updated accordingly.

$\mathsf{Send}_J(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$: Assume the user with current identity (ID_I, ι_I) is joining the group. If Ψ is an index K, $\mathcal{C}$ outputs $(ID_i, \iota_i, r_K, u_K, \{z_{K,j}\}_{j\in\{1,\ldots,n\},j\neq K})$ which is generated in the same way as the tuple in the $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$ query. Note that it is required that the K-th identity in $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ be $\emptyset$. Else if $\mathcal{U}_i$ is not the group manager, a new instance of $\mathcal{U}_i$ $\Pi^{\pi'}_{\mathcal{U}_i}$ is initiated. $\mathcal{C}$ sets $\mathsf{ms}^{\pi'}_{\mathcal{U}_i} = \mathsf{ms}^{\pi}_{\mathcal{U}_i}$, sets the K-th tuple in $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ to be Ψ, sets $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and the K-th current identity in $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ to be (ID_I, ι_I). Else $\mathcal{U}_i$ is a group manager, and a new instance of $\mathcal{U}_i$ $\Pi^{\pi'}_{\mathcal{U}_i}$ is initiated. $\mathcal{C}$ responds with $\{(ID_i, \iota_i, r_i, u_i, z_{i,K})\}_{i\in\{1,\ldots,n\},i\neq K})$, which is recorded in $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$, sets $\mathsf{ms}^{\pi'}_{\mathcal{U}_i} = \mathsf{ms}^{\pi}_{\mathcal{U}_i}$, sets the K-th tuple in $\mathsf{ms}^{\pi'}_{\mathcal{U}_i}$ to be Ψ, sets $\mathsf{pid}^{\pi'}_{\mathcal{U}_i} = \mathsf{pid}^{\pi}_{\mathcal{U}_i}$ and the K-th current identity in $\mathsf{pid}^{\pi'}_{\mathcal{U}_i}$ to be (ID_I, ι_I).

$\mathsf{Ek.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: Assume $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \{ID_1, \iota_1; \ldots; ID_n, \iota_n\}$, $\mathsf{isid}^{\pi}_{\mathcal{U}_i} = \mathsf{isid}$, and $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma_1, \ldots, \sigma_n\}$, where σ_l equals

$$(ID_l, \iota_l, r_l, u_l, \{z_{l,j}\}_{j\in\{1,\ldots,n\},j\neq l}) \text{ or } (ID_l, \iota_l, r_l, u_l, z_{l,i}).$$

If $\mathsf{state}^{\pi}_{\mathcal{U}_i} = \textit{accepted}$ and $\Delta = 1$, $\mathcal{C}$ computes $w = \prod_{l=1}^{n} r_l, y = \prod_{l=1}^{n} u_l, \varpi_l = H_4(\mathsf{isid}, ID_l, \iota_l, r_l, u_l)$, $\Omega = \hat{e}(\prod_{l=1}^{n}(H_1(ID_l, \iota_l, 0)H_1(ID_l, \iota_l, 1)^{\varpi_l}), g_{pub})\hat{e}(H_2(\mathsf{isid}), y)$ and returns (w, Ω); otherwise, it returns $null$.

$\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: $\mathcal{C}$ first finds $coin^{\pi}_{\mathcal{U}_i}$. If $coin^{\pi}_{\mathcal{U}_i} = 1$, $\mathcal{C}$ aborts (E3); else if $\Pi^{\pi}_{\mathcal{U}_i}$ is not *accepted*, returns $null$; else, recovers the corresponding $z_{i,i}$ from S^{list} and $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma_1, \ldots, \sigma_n\}$, outputs $d_i = \prod_{l=1}^{n} z_{l,i}$.

$\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: 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 the session ID is sid^*, the initial session ID is isid^*, $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \{ID^{\pi_1}_{\mathcal{U}^*_1}; \ldots; ID^{\pi_n}_{\mathcal{U}^*_n}\}$, $\mathsf{ms}^{\pi}_{\mathcal{U}_i} = \{\sigma^*_1, \ldots, \sigma^*_n\}$, where $ID^{\pi_l}_{\mathcal{U}^*_l} = (ID^*_l, \iota^*_l)$, σ^*_l equals $(ID^*_l, \iota^*_l, r^*_l, u^*_l, \{z^*_{l,j}\}_{j\in\{1,\ldots,n\},j\neq l})$ or $(ID^*_l, \iota^*_l, r^*_l, u^*_l, z^*_{l,i})$. $\mathcal{C}$ does the following:

1) For $1 \leq l \leq n$, recover $coin^{\pi_l}_{\mathcal{U}^*_l}$. If $coin^{\pi_1}_{\mathcal{U}^*_1} = \ldots = coin^{\pi_n}_{\mathcal{U}^*_n} = 1$ and $\Delta = 1$, turn to Step 2; otherwise, abort (E4).
2) Recover $(\mathsf{isid}^*, \gamma^*, v^*)$ from H_2^{list} and $(\eta^*_1, \theta^*_1), \ldots, (\eta^*_n, \theta^*_n)$ from S^{list}. Since $\Delta = 1$ and the underlying stateful IBBMS is strongly unforgeable, with overwhelming probability $1 - 2\epsilon'$, we have that $\mathsf{ek}^{\pi}_{\mathcal{U}_i} = (w^*, \Omega^*)$, i.e., $(g^{\sum_{l=1}^{n}\eta^*_l}, \hat{e}(\prod_{l=1}^{n}(id^*_{l,\iota_l,0} id^{*\,\varpi^*_l}_{l,\iota_l,1}), g^{\kappa})\hat{e}(g_1 g^{\gamma^*}, y^*))$, where $id^*_{l,\iota_l,0} = H_1(ID^*_l, \iota^*_l, 0), id^*_{l,\iota_l,1} = H_1(ID^*_l, \iota^*_l, 1), \varpi^*_l = H_4(\mathsf{isid}^*, ID^*_l, \iota^*_l, r^*_l, u^*_l), y^* = \prod_{l=1}^{n} u^*_l = g^{\sum_{i=1}^{n}\theta^*_i} g_k$ and ϵ' is the advantage of $\mathcal{A}$ in forging a valid IBBMS.
3) Set $c_1 = h, c_2 = h^{\sum_{l=1}^{n}\eta^*_l}$, randomly choose $\psi \in \{0,1\}^{l_0}$, and compute $c_3 = m_b \oplus \psi$, where $b \in \{0,1\}$.
4) 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_5 on $\hat{e}(\prod_{l=1}^{n}(id^*_{l,\iota_l,0} id^{*\,\varpi^*_l}_{l,\iota_l,1}), h^{\kappa})\hat{e}(h^{\gamma^*}, y^*)\hat{e}(g_1, h^{\sum_{i=1}^{n}\theta_i})\hat{e}(h, g^{\alpha^{k+1}})$.

Response: Once $\mathcal{A}$ finishes querying and returns its guess $b' \in \{0,1\}$, $\mathcal{C}$ randomly chooses a tuple (Φ_i, ϱ_i) from H_5^{list} and returns the value $\Phi_i \cdot (\hat{e}(\prod_{l=1}^{n}(id^*_{l,\iota_l,0} id^{*\,\varpi^*_l}_{l,\iota_l,1}), h^{\kappa})\hat{e}(h^{\gamma^*}, y^*)\hat{e}(g_1, h^{\sum_{i=1}^{n}\theta_i}))^{-1}$ as the response to the k-BDHE challenge.

If $\mathcal{C}$ does not abort, the above simulations of all queries are valid and the answers are uniformly distributed. Hence, the adversary cannot find any inconsistency between the simulation and the real world. Therefore, $\Pr[b = b'] \geq \epsilon$. It remains to determine the probability that $\mathcal{C}$ outputs the required Φ_i. It is easy to see that $\mathcal{C}$ will abort if E3 or E4 happen. We have to calculate $\Pr[\neg\mathsf{E3} \wedge \neg\mathsf{E4}]$.

It can be seen that $\Pr[\neg\mathsf{E3}] \geq (1-\delta)^{q_{DK}}$, $\Pr[\neg\mathsf{E4} \geq \delta^n(1-2\epsilon')$. Since these probabilities are independent, the overall probability that $\mathcal{C}$ does not abort is $\delta^n(1-\delta)^{q_{DK}}(1-2\epsilon')$. Hence, we have $\Pr[\neg\mathsf{E3} \wedge \neg\mathsf{E4}] \geq \delta^n(1-\delta)^{q_{DK}}(1-2\epsilon') \geq \frac{n^n(1-2\epsilon')}{((q_{DK}+1)e)^n}$.

In conclusion, we have that the probability for $\mathcal{C}$ to solve the k-BDHE problem is at least $\frac{n^n(1-2\epsilon')}{((q_{DK}+1)e)^n}\epsilon$. □

6 Performance Evaluation

6.1 Comparison

We compare our dynamic protocol with the unauthenticated AGKA protocol in [20] and the IBAAGKA protocols in [21], [23], [24]. We note that the other protocols are static ones which do not support user leave or join. In the sequel, only costly operations are considered and the operations that can be pre-computed are omitted.

Table 3 compares our dynamic protocol with the other protocols regarding transmission cost, where $P_1, P_2, P_{ID}, P_m, P_{sig}, \iota$ denote the binary length of an element in $\mathbb{G}_1$, $\mathbb{G}_2$, an identity, a message, an identity-based signature and the index of a user's private key, respectively. The table shows that our protocol has a slightly lower transmission overhead than the ones in [20], [21], [23] and has comparable transmission overhead with the one in [24] in the Agreement stage if we consider an identity of length 160 bits. As to

TABLE 3: Transmission Overhead

Protocols	Agreement/Join/Leave	Ciphertext	Security
AGKA in [20]	$nP_1 + P_2$/static/static	$2P_1 + P_2$	unauthenticated
AGKA in [21], [23]	$nP_1 + P_{sig} + P_{ID}$/static/static	$2P_1 + P_m$	PFS
AGKA in [24]	$(n+1)P_1 + P_{ID}$/static/static	$2P_1 + P_m$	KEF§
Proposed protocol	$(n+1)P_1 + P_{ID} + \iota$	$2P_1 + P_m$	KEF

§: The formal security proof is not provided.

the ciphertext size, all the protocols have the same size, assuming that the plaintexts are of the same size. Further, in our protocol, if a regular user joins/leaves the group, the message transmitted by the new member/group manager has the same size as the one by a member in the Agreement stage.

The comparison of computational overhead is shown in Table 4, where $\tau_{\hat{e}}$ denotes the time to compute a bilinear map, $\tau_{\mathbb{G}_1}$ denotes the time to compute a scalar exponentiation in $\mathbb{G}_1$, τ_H is the time to compute a MapToPoint hash [24], $\tau_{\mathbb{G}_2}$ is the time to compute a scalar exponentiation in $\mathbb{G}_2$, τ_{sg} is the time to generate an identity-based signature (IBS) and τ_{sv} denotes the verification time of an IBS. For simplicity, in the following, we only consider the cost for a participant to generate the group encryption key while we omit the cost for a sender to generate the group encryption key. From Table 1, it is easy to see that our protocol has comparable efficiency in stages Agreement, Enc.Key.Gen and Dec.Key.Gen with those in [20], [21], [23], [24]. Our protocol is as efficient as the protocols in [21], [23], [24] and has similar efficiency as the protocol in [20] in stages Enc and Dec. If a user wants to join the group, the computational cost is the same as that of a user in the Agreement stage. However, if a user leaves the group, the cost is 0 for all the users in the group, since all the operations can be pre-computed.

6.2 Simulation

We show the practical performance of our protocol through a simulation. The simulation was run on a Linux machine using AMD FX-8120 at 3.1GHz. The cryptographic algorithms were implemented using the pairing-based cryptography (PBC) library. In our simulation, a type A curve was chosen, the security parameter ℓ chosen was 160, the protocol participants ranged from 3 to 100. The computations which can be pre-computed were omitted, e.g., the computations at Steps 1 and 2 in Key Establishment stage. The efficiency of our dynamic protocol is mainly dominated by the last 6 stages, since Setup only needs to be run once when the system is initialized and Extract only requires to be run once when a user joins the system. Hence, in the sequel, we only evaluate the efficiency of the last 6 stages of our protocol.

Figure 2 shows the relationship between the number of participants and the execution time of each stage of our protocol. From this figure, we can see that the time costs for a outsider and a group member to generate a group encryption key are almost the same and higher than those of other stages. They grow linearly as the number of participants grows. However, the time costs are still not high. When the group size is 100, they are less than 1 second. We note that the efficiency of the Enc.Key.Gen stage will not significantly

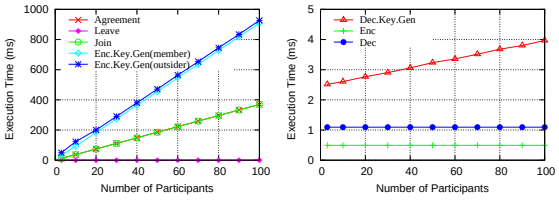


Fig. 2: Average execution time

affect the efficiency of the whole protocol, because the group members do not need to run Enc.Key.Gen frequently. The time cost of Agreement is the same as that of Join. The time cost of Leave is 0. This is because all the operations in this stage can be pre-computed. The execution time for a participant to generate a decryption key grows linearly with the number of participants. This time cost ranges from 2.52 ms to 3.97 ms and it is much smaller than the one of the group encryption key generation. This is because the most costly operations are performed in the Enc.Key.Gen stage. The time costs to generate a ciphertext and decrypt a ciphertext are constant regardless of the number of protocol participants. They are 0.49 ms and 1.10 ms, respectively.

7 Conclusion

We have defined the security model for dynamic IBAAGKA protocols, in which an attacker is allowed to learn the master secret of the KGC. A one-round dynamic IBAAGKA protocol is proposed and proven secure in our model under the k-BDHE assumption. It offers secrecy and known-key security, and it does not suffer from the key escrow problem. Therefore, not even the KGC can decrypt the ciphertexts sent to a group.

8 Acknowledgment and Disclaimer

This work was supported in part by the the NSF of China under Grants 61202465, 61021004, 11061130539, 61103222, 61173154, 61003214; EU FP7 under projects "Dw-B" and "Inter-Trust"; the Spanish Government under project TIN2011-27076-C03-01; the Government of Catalonia under grant SGR2009-1135; the Shanghai NSF under Grant No. 12ZR1443500; the Shanghai Chen Guang Program (12CG24); the Fundamental Research Funds for the Central Universities of China.

Appendix

Let $\mathcal{C}$ be a CDH challenger who is given an instance $(g, g^{\alpha}, g^{\beta})$. We show that if an adversary $\mathcal{A}$ can break the proposed IBBMS scheme under an adaptive chosen-message

TABLE 4: Computational Overhead

Protocols	Agreement/Join	Leave	Enc.Key.Gen	Dec.Key.Gen	Enc	Dec
AGKA in [20]	$1\tau_{\hat{e}} + n\tau_{\mathbb{G}_1}$/static	static	-$^{\flat}$	$1\tau_{\mathbb{G}_1}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_2}$
AGKA in [21], [23]†	$(n+1)\tau_{\mathbb{G}_1} + n\tau_H + 1\tau_{sg}$/static	static	$1\tau_{\hat{e}}$	$2\tau_{\hat{e}} + n\tau_{sv}^{\ddagger}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$
AGKA in [24]	$(n+4)\tau_{\mathbb{G}_1} + (n+1)\tau_H$/static	static	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$	$2\tau_{\hat{e}}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$
Proposed protocol	$(n+4)\tau_{\mathbb{G}_1} + (n+1)\tau_H$	0	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$	$2\tau_{\hat{e}}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$

$\flat$: Only lightweight operations are required.
$\dagger$: An identity-based signature is used to secure the system.
$\ddagger$: The verification of n signatures can be performed in either stage Enc.Key.Gen or Dec.Key.Gen.

attack, then $\mathcal{C}$ can use $\mathcal{A}$ to solve the CDH problem, i.e., to compute $g^{\alpha\beta}$, which is hard.

Initialize: $\mathcal{C}$ sets $g_{pub} = g^{\alpha}$, selects $\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, , \hat{e}, g, g_{pub}, H_1 \sim H_4)$ and gives Υ to $\mathcal{A}$.

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

H_1 queries: $\mathcal{C}$ maintains an initially empty list H_1^{list}. On input $(ID_i, j), j \in \{0,1\}$, $\mathcal{C}$ does the following

- If there is a tuple $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ in H_1^{list}, return $id_{i,j}$ as the answer.
- Else flip a coin $coin_{1,i} \in \{0,1\}$ that yields 1 with probability δ and 0 with probability $1-\delta$ and proceed as follows:
  - If $coin_{1,i} = 0$, pick $\mu'_{i,0}, \mu'_{i,1} \in \mathbb{Z}_q^*$, set $\mu_{i,0} = \mu_{i,1} = 0, id_{i,0} = g^{\mu'_{i,0}}, id_{i,1} = g^{\mu'_{i,1}}, s_{i,0} = g_{pub}^{\mu'_{i,0}}$, $s_{i,1} = g_{pub}^{\mu'_{i,1}}$, add $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ to H_1^{list} and return $id_{i,j}$.
  - Else, choose $\mu_{i,0}, \mu_{i,1}, \mu'_{i,0}, \mu'_{i,1} \in Z_q^*$, set $id_{i,0} = g^{\beta\mu_{i,0}} g^{\mu'_{i,0}}, id_{i,1} = g^{\beta\mu_{i,1}} g^{\mu'_{i,1}}, s_{i,0} = s_{i,1} = null$, add $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ to H_1^{list} and respond with $id_{i,j}$.

H_2 queries: $\mathcal{C}$ keeps an initially empty list H_2^{list}. On input $info_i$, if there is a tuple $(info_i, \gamma_i, v_i)$ in H_2^{list}, $\mathcal{C}$ returns v_i; else randomly selects $\gamma_i \in \mathbb{Z}_q^*$, sets $v_i = g^{\gamma_i}$, adds $(info_i, \gamma_i, v_i)$ to H_2^{list} and returns v_i.

H_3 queries: $\mathcal{C}$ keeps an initially empty list H_3^{list}. On input $(info_i, m_i)$, $\mathcal{C}$ does the following:

- If there is a tuple $(info_i, m_i, \zeta_i, f_i, coin_{3,i})$ in H_3^{list}, return f_i as the answer.
- Else if $info_i$ appears in tuple $(info'_i, m'_i, \zeta'_i, f'_i, coin'_{3,i})$, set $coin_{3,i} = coin'_{3,i}$, randomly select $\zeta_i \in \mathbb{Z}_q^*$. If $coin_{3,i} = 0$, set $f_i = g^{\zeta_i} g^{\alpha}$, add $(info_i, m_i, \zeta_i, f_i, coin_{3,i})$ to H_3^{list} and return f_i as the answer. Else compute $f_i = g^{\zeta_i}$, add $(info_i, m_i, \zeta_i, f_i, coin_{3,i})$ to H_3^{list} and return f_i as the answer.
- Else flip a coin $coin_{3,i}$ that yields 1 with probability δ and 0 with probability $1-\delta$, randomly select $\zeta_i \in \mathbb{Z}_q^*$. If $coin_{3,i} = 0$, set $f_i = g^{\zeta_i} g^{\alpha}$, add $(info_i, m_i, \zeta_i, f_i, coin_{3,i})$ to H_3^{list} and return f_i as the answer. Else compute $f_i = g^{\zeta_i}$, add $(info_i, m_i, \zeta_i, f_i, coin_{3,i})$ to H_3^{list} and return f_i as the answer.

H_4 queries: $\mathcal{C}$ keeps an initially empty list H_4^{list}. On input $(info_i, ID_i, r_i, u_i)$, if there is a tuple $(info_i, ID_i, r_i, u_i, \varpi_i)$ in H_4^{list}, $\mathcal{C}$ returns ϖ_i as the answer; else, $\mathcal{C}$ recovers the tuple $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ from H_1^{list}, randomly chooses $\varpi_i \in \mathbb{Z}_q^* \setminus \{-\frac{\mu_{i,0}}{\mu_{i,1}}\}$, adds $(info_i, ID_i, r_i, u_i, \varpi_i)$ to H_4^{list} and responds with ϖ_i.

BM.Extract queries: On input an identity ID_i, $\mathcal{C}$ first makes a H_1 query on ID_i, then it recovers $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ from H_1^{list}. If $coin_{1,i} = 0$, $\mathcal{C}$ returns $(s_{i,0}, s_{i,1})$ as the answer; otherwise, it aborts (E1).

Sign queries: The challenger $\mathcal{C}$ keeps an initially empty list S^{list}. On input $(ID_i, info_l, m_1, \ldots, m_{t_i})$, the challenger $\mathcal{C}$ first submits $(ID_i, 0)$ to H_1 if ID_i never appears in H_1^{list} and issues a H_2 query on $info_l$, H_3 queries on $(info_l, m_j)$ for $1 \le j \le t_i$, if those queries have never been issued; then $\mathcal{C}$ recovers $(ID_i, \mu_{i,0}, \mu'_{i,0}, \mu_{i,1}, \mu'_{i,1}, id_{i,0}, id_{i,1}, s_{i,0}, s_{i,1}, coin_{1,i})$ from H_1^{list}, $(info_l, \gamma_l, v_l)$ from H_2^{list} and $(info_l, m_j, \zeta_j, f_j, coin_{3,j})$ from H_3^{list} for $1 \le j \le t_i$. Finally, $\mathcal{C}$ does the following:

- If the query with same input was queried before, the same answer is returned.
- Else if $coin_{1,i} = 1$, do the following:
  - If $coin_{3,j} = 0$, do the following steps:
    1) Randomly select $\eta_i \in \mathbb{Z}_q^*, \varpi_i \in \mathbb{Z}_q^* \setminus \{-\frac{\mu_{i,0}}{\mu_{i,1}}\}$ and $u_i \in \mathbb{G}_1$.
    2) Compute $r_i = g^{\eta_i} g^{-\beta(\mu_{i,0} + \varpi_i \mu_{i,1})}$. If $(info_l, ID_i, r_i, u_i, \varpi_i) \in H_4^{list}$, go to Step 1; otherwise, set $H_4(info_l, ID_i, r_i, u_i) = \varpi_i$ and add $(info_l, ID_i, r_i, u_i, \varpi_i)$ to H_4^{list}.
    3) Compute $z_{i,j} = r_i^{\zeta_j} u_i^{\gamma_l} g_{pub}^{\mu'_{i,0} + \varpi_i \mu'_{i,1} + \eta_i}$ for $1 \le j \le t_i$.
  - Else if $coin_{3,j} = 1$, do the following steps:
    1) If there is a tuple $(info_l, ID_i, r_i, u_i, \varpi_i)$ in H_4^{list} such that $\varpi_i = -\frac{\mu_{i,0}}{\mu_{i,1}}$, abort (E2); otherwise, go to Step 2.
    2) Randomly select $r_i, u_i \in \mathbb{G}_1$.
    3) If $(info_l, ID_i, r_i, u_i)$ appears in H_4^{list}, go to Step 2; otherwise, set $\varpi_i = H_4(info_l, ID_i, r_i, u_i) = -\frac{\mu_{i,0}}{\mu_{i,1}}$ and add $(info_l, ID_i, r_i, u_i, \varpi_i)$ to H_4^{list}.
    4) Compute $z_{i,j} = r_i^{\zeta_j} u_i^{\gamma_l} g_{pub}^{(\mu'_{i,0} + \varpi_i \mu'_{i,1})}$ for $1 \le j \le t_i$.

  Finally, return $(r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,t_i\}})$, add $(info_l, ID_i, \{m_j\}_{j \in \{1,\ldots,t_i\}}, r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,t_i\}})$ to S^{list}.
- Else ($coin_{1,i} = 0$) use the Sign algorithm to generate and output a batch signature $(r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,t_i\}})$, since the private key corresponding to ID_i

is known to $\mathcal{C}$. Add the tuple $(info_l, ID_i, \{m_j\}_{j\in\{1,\ldots,t_i\}}, r_i, u_i, \{z_{i,j}\}_{j\in\{1,\ldots,t_i\}})$ to S^{list}.

We note that in the above queries, if $\mathcal{A}$ asks a batch signature query with a previously used state information but different messages as input, $\mathcal{C}$ returns $null$.

Forgery: Finally, $\mathcal{A}$ outputs x identities $(ID_1^*, \ldots, ID_x^*)$, a piece of state information $info^*$, a message m^* and a multi-signature $\sigma^* = (r_1^*, \ldots, r_x^*, u_1^*, \ldots, u_x^*, d^*)$, where σ^* is a valid multi-signature on m^* under $info^*$ and $(ID_1^*, \ldots, ID_x^*)$.

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

1) Submit $(ID_i^*, 0)$ to H_1 and $(info^*, ID_i^*, r_i^*, u_i^*)$ to H_4 for $1 \le i \le x$, submit $info^*$ to H_2, $(info^*, m^*)$ to H_3 if those queries have never been issued.
2) Recover $(ID_i^*, \mu_{i,0}^*, \mu_{i,0'}^*, \mu_{i,1}^*, \mu_{i,1'}^*, id_{i,0}^*, id_{i,1}^*, s_{i,0}^*, s_{i,1}^*, coin_{1,i}^*)$ from H_1^{list} and $(info^*, ID_i^*, r_i^*, u_i^*, \varpi_i^*)$ from H_4^{list} for $1 \le i \le x$, find $(info^*, \gamma^*, v^*)$ in H_2^{list}, and $(info^*, m^*, \zeta^*, f^*, coin_3^*)$ in H_3^{list}.

For $\mathcal{C}$ to solve the given instance of the CDH problem, it requires that $coin_3^* = 1$ and none of the identities in $\{ID_1^*, \ldots, ID_x^*\}$ has been submitted during the BM.Extract queries (i.e., $coin_{1,i}^* = 1$ for $1 \le i \le x$) and at least one of $\varpi_i^* \neq -\frac{\mu_{i,0}^*}{\mu_{i,1}^*}$ for $1 \le i \le x$; otherwise, $\mathcal{C}$ aborts. If $\mathcal{C}$ does not abort, by our simulation, we have $id_{i,0}^* = g^{\beta\mu_{i,0}^*} g^{\mu_{i,0}'^*}$, $id_{i,1}^* = g^{\beta\mu_{i,1}^*} g^{\mu_{i,1}'^*}$, $v^* = g^{\gamma^*}$ and $f^* = g^{\zeta^*}$. Since σ^* should be valid, we have

$$\begin{aligned}
& \hat{e}(d^*, g) \\
= \; & \hat{e}(\prod_{i=1}^{x} id_{1,0}^* id_{1,1}^{*\,\varpi_i^*}, g_{pub})\hat{e}(v^*, \prod_{i=1}^{x} u_i^*)\hat{e}(f^*, \prod_{i=1}^{x} r_i^*) \\
= \; & \hat{e}(g^{\alpha\beta\sum_{i=1}^{x}(\mu_{1,0}^*+\varpi_1^*\mu_{1,1}^*)} g_{pub}^{\sum_{i=1}^{x}(\mu_{i,0}'^*+\varpi_i^*\mu_{i,0}'^*)}, g) \times \\
& \hat{e}(g, (\prod_{i=1}^{x} u_i^*)^{\gamma^*})\hat{e}(g, (\prod_{i=1}^{x} r_i^*)^{\zeta^*}).
\end{aligned}$$

Since we require that at least one of $\varpi_i^* \neq -\frac{\mu_{i,0}^*}{\mu_{i,1}^*}$, by the simulation, the probability of $\sum_{i=1}^{x}(\mu_{i,0}^* + \varpi_i^*\mu_{i,1}^*) = 0$ is negligible. Further, for randomly chosen $\mu_{i,0}'^*, \mu_{i,1}'^*$, the probability of $\sum_{i=1}^{x}(\mu_{i,0}'^* + \varpi_i^*\mu_{i,0}'^*) = 0$ is $\frac{1}{2^q}$, and will be omitted below. Hence, we have the solution of the CDH problem $g^{\alpha\beta} = \vartheta^{(\sum_{i=1}^{x}(\mu_{i,0}^*+\varpi_i^*\mu_{i,1}^*))^{-1}}$, where $\vartheta = d^*(g_{pub}^{\sum_{i=1}^{x}(\mu_{i,0}'^*+\varpi_i^*\mu_{i,0}'^*)}(\prod_{i=1}^{x} u_i^*)^{\gamma^*}(\prod_{i=1}^{x} r_i^*)^{\zeta^*})^{-1}$.

To complete the proof, we need 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 $info^*$ and $(ID_1^*, \ldots, ID_x^*)$.
- $\mathcal{E}3$: $\mathcal{E}2$ occurs, and, $coin_3^* = 1$, and, $coin_{1,i}^* = 1$ for $1 \le i \le x$, and $\sum_{i=1}^{x}(\mu_{i,0}^* + \varpi_i^*\mu_{i,1}^*) \neq 0$.

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

In the above simulation, $\mathcal{C}$ will abort if E1 or E2 happen. It is easy to see that $\Pr[\neg\mathcal{E}1] = (1-\delta)^{q_E}$. $\mathcal{E}2$ will not happen if in all the H_3 queries, only one state information leads to $coin_{3,i} = 1$. We have $\Pr[\neg \mathsf{E}\ 2] \ge q_{H_3}\delta(1-\delta)^{q_{H_3}-1}$. Therefore, we have $\Pr[\mathcal{E}1] \ge q_{H_3}\delta(1-\delta)^{q_E+q_{H_3}-1}$.

If $\mathcal{C}$ does not abort, $\mathcal{C}$ will forge a valid and nontrivial multi-signature with probability ϵ'. Hence, we have $\Pr[\mathcal{E}2|\mathcal{E}1] \ge \epsilon'$.

For $\mathcal{E}3$, by our simulation, $\varpi_i^* = -\frac{\mu_{i,0}^*}{\mu_{i,1}^*}$ happens unless $\mathcal{A}$ issues a Sign query. Since we require that the multi-signature is not generated by using all the batch signatures output by calling the Sign queries with $(ID_i^*, info^*, m_1, \ldots, m_{\mathcal{I}}, \ldots, m_t)$ as input, for $ID_i^* \in \{ID_1^*, \ldots, ID_x^*\}$ (where $m_{\mathcal{I}} = m^*$ and $\mathcal{I}$ defines the index of the message), at least one of $\varpi_i^* \neq -\frac{\mu_{i,0}^*}{\mu_{i,1}^*}$. Therefore, the probability for $\sum_{i=1}^{x}(\mu_{i,0}^* + \varpi_i^*\mu_{i,1}^*) = 0$ is $\frac{1}{2^q}$, which is negligible and will be omitted. We have $\Pr[\mathcal{E}3|\mathcal{E}1 \wedge \mathcal{E}2] = \delta^{x+1}$.

Totally, we have $\Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3] \ge (\frac{x+2}{q_E+q_{H_3}+x+1})^{x+2}\frac{q_{H_3}}{e^{x+2}}\epsilon'$.

REFERENCES

[1] M. Au, J. Liu, W. Susilo and J. Zhou, "Realizing Fully Secure Unrestricted ID-Based Ring Signature in the Standard Model Based on HIBE", *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp.1909-1922 2013

[2] M. Bellare and P. Rogaway, "Entity authentication and key distribution", *Proc. 13th Ann. Int. Cryptology Conf. (CRYPTO93)*, pp.232-249 1994

[3] D. Boneh, X. Boyen and E. J. Goh, "Hierarchical identity based encryption with constant size ciphertext", *Proc. 24th Ann. Int. Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'05)*, pp.440-456 2005

[4] D. Boneh and A. Silverberg, "Applications of multilinear forms to cryptography", *Contemporary Mathematics*, vol. 324, no. 1, pp.71-90 2003

[5] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system", *Proc. Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT'94)*, pp.275-286 1995

[6] L. Chen and C. Kudla, "Identity based authenticated key agreement protocols from pairings", *Proc. 16th IEEE Comput. Security Foundations Workshop (CSFW'03)*, pp.219-233 2003

[7] L. Chen, Z. Cheng and N. P. Smart, "Identity-based key agreement protocols from pairings", *Int. J. Inf. Security*, vol. 6, no. 4, pp.213-241 2007

[8] K. Y. Choi, J. Y. Hwang and D. H. Lee, "Efficient ID-based group key agreement with bilinear maps", *Proc. 7th Int. Workshop on Theory and Practice in Public Key Cryptography (PKC'04)*, pp.130-144 2004

[9] A. Fiat and M. Naor, "Broadcast encryption (Extended abstract)", *Proc. 13th Ann. Int. Cryptology Conf. (CRYPTO93)*, pp.480-491 1994

[10] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes" *Proc. 19th Ann. Int. Cryptology Conf. (CRYPTO'99)*, pp.537-554 1999

[11] S. Garg, C. Gentry and S. Halevi, "Candidate multilinear maps from ideal lattices and applications", *Proc. 32nd Ann. Int. Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'13)* pp. 1-17 2013

[12] C. Gentry and B. Waters, "Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts)", *Proc. 28th Ann. Int. Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'09)*, pp.171-188 2009

[13] I. Ingemarsson, D. T. Tang and C. K. Wong, "A conference key distribution system", *IEEE Trans. Inf. Theory*, vol. 28, no. 5, pp.714-720 1982

[14] L. Lai, Y. Liang and H. Poor, "A Unified Framework for Key Agreement Over Wireless Fading Channels", *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp.480-490 2012

[15] X. Lv, H. Li and B. Wang, "Authenticated asymmetric group key agreement based on certificateless cryptosystem", *Int. J. Comput. Mathematics*, vol. 91, no. 3, pp.447-460 2014

[16] X. Lv, Y. Mu and H. Li, "Non-Interactive Key Establishment for Bundle Security Protocol of Space DTNs", *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 1, pp.5-13 2014

[17] K. C. Reddy and D. Nalla, "Identity based authenticated group key agreement protocol", *Proc. 3rd Int. Conf. Cryptology in India (INDOCRYPT'02)*, pp.215-233 2002

[18] M. Steiner, G. Tsudik and M. Waidner, "Key agreement in dynamic peer groups", *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp.769-780 2000

[19] G. Wei, X. Yang and J. Shao, "Efficient certificateless authenticated asymmetric group key agreement protocol", *KSII Trans. Internet Inf. Syst.*, vol. 6, no. 12, pp.3352-3365 2012

[20] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, "Asymmetric group key agreement", *Proc. 28th Ann. Int. Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'09)*, pp.153-170 2009

[21] L. Zhang, Q. Wu, B. Qin and J. Domingo-Ferrer, "Identity-based authenticated asymmetric group key agreement protocol", *Prof. 16th Annual International Computing and Combinatorics Conference (COCOON 2010)* vol. 6196, pp.510-519 2010

[22] L. Zhang, Q. Wu, B. Qin, H. Deng, J. Liu and W. Shi, "Provably Secure Certificateless Authenticated Asymmetric Group Key Agreement", *Prof. 10th International Conference on Information Security Practice and Experience (ISPEC 2014)*, vol. 8434, pp.496-510 2014

[23] L. Zhang, Q. Wu, B. Qin and J. Domingo-Ferrer, "Provably secure one-round identity-based authenticated asymmetric group key agreement protocol", *Information Sciences*, vol. 181, no. 19, pp.4318-4329 2011

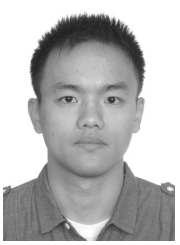
[24] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, S. Chow and W. Shi, "Secure One-to-Group Communications: Escrow-Free ID-Based Asymmetric Group Key Agreement", *Proc. 9th China International Conference on Information Security and Cryptology (INSCRYPT 2013)* vol. 8567, pp. 239-254 2014

Lei Zhang received his Ph.D. degree from Universitat Rovira i Virgili in 2010. He is an Associate Research Fellow in East China Normal University. Before this, he had been a Postdoctoral Researcher with Universitat Rovira i Virgili. His fields of activity are information security, cryptography, data privacy, and network security. He has been a holder/co-holder of 5 China/Spain funded projects. He has authored about 50 publications. He has served in the program committee of several international conferences in information security and privacy. He is a member of IEEE.

Qianhong Wu received his Ph.D. from Xidian University in 2004. Since then, he has been with Wollongong University as an associate research fellow, with Wuhan University as an associate professor, and with Universitat Rovira i Virgili as a research director. He is currently a professor in Beihang University. His research interests include cryptography, information security and privacy. He has been a holder/co-holder of 8 China/Australia/Spain funded projects. He has authored 7 patents and over 110 publications. He has served in the program committee of several international conferences in information security and privacy.

Josep Domingo-Ferrer is a Distinguished Professor of Computer Science and ICREA-Acadèmia Researcher at Universitat Rovira i Virgili, Catalonia, where he holds the UNESCO Chair in Data Privacy. His research interests include data privacy, data security and cryptographic protocols. He received MSc and PhD degrees from the Autonomous University of Barcelona, Catalonia; he also holds an MSc in Mathematics. He is an IEEE Fellow and an Elected Member of Academia Europaea and he has received a Google Research Award. He has authored over 340 publications and 5 patents, and his current H-index is 42.

Bo Qin received her Ph.D. degree from Xidian University in 2008. Since then, she has been with Xi'an University of Technology as a lecturer and with Universitat Rovira i Virgili as a postdoctoral researcher. She is currently a lecturer in the Renmin University. Her research interests include data security and privacy. She has been a holder/co-holder of 5 China/Spain funded projects. She has authored over 60 publications and served in the program committee of several international conferences in information security.



Zheming Dong is currently a master student in Software Engineering Institute, East China Normal University, China. His fields of activity are information security, data privacy, network security, wireless sensor network.