Secure One-to-Group Communications Escrow-Free ID-Based Asymmetric Group Key Agreement

Lei Zhang[1], Qianhong Wu[2(✉)], Josep Domingo-Ferrer [3], Bo Qin[4], Sherman S.M. Chow[5], and Wenchang Shi[4]

[1] Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai, China
leizhang@sei.ecnu.edu.cn
[2] School of Electronic and Information Engineering, Beihang University, Beijing, China
qianhong.wu@buaa.edu.cn
[3] UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia, Spain
josep.domingo@urv.cat
[4] School of Information, Renmin University of China, Beijing, China
{bo.qin,wenchang}@ruc.edu.cn
[5] Department of Information Engineering, Chinese University of Hong Kong, Sha Tin, Hong Kong
sherman@ie.cuhk.edu.hk

Abstract. Group key agreement (GKA) is widely employed for secure group communications. Yet there is an increasing demand for secure one-to-group communications in distributed computing applications. Asymmetric group key agreement (AGKA) is a handy tool to answer this need. In AGKA, a group of members can establish a group public key while each member has a different secret key. Any sender can encrypt under this group key such that any of the members who hold the secret key can decrypt. This paper proposes an identity-based AGKA protocol which is secure against active attackers, with an emphasis on optimal round efficiency, sender dynamics, and escrow freeness. The last feature offers security of the previously established ciphertexts even when either all the involved participants or the key generation center of the identity-based cryptosystem are compromised. The proposed protocol is shown to be secure under the k-Bilinear Diffie-Hellman exponent assumption in the random oracle model. Regarding performance, our protocol is comparable to the state-of-the-art AGKA protocols.

Keywords: Communication security · Key management · Identity-based cryptography · Asymmetric group key agreement

D. Lin et al. (Eds.): Inscrypt 2013, LNCS 8567, pp. 239–254, 2014.
DOI: 10.1007/978-3-319-12087-4_15

1 Introduction

In ubiquitous computing applications such as wireless mesh networks and mobile *ad hoc* networks, there is a need to efficiently and securely broadcast to a remote cooperative group. A popular approach to secure group communications is to exploit group key agreement (GKA) [5]. 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. Thus, conventional GKA protocols are *sender restricted* in the sense 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 then run a GKA protocol. To see the limitations, let us consider the following scenarios.

Scenario 1. A group of users in different time zones would like to discuss on some sensitive topics over an untrusted medium, e.g., via a social network service provider.

Scenario 2. One or more soldiers may want to securely report to a group of tactical units.

Up to now, most existing efficient GKA protocols need at least two rounds [12,19]. In Scenario 1, all the users have to stay online to finish the protocol before they can wait for encrypted contents, which is a prohibitive way for users in different time zones. In Scenario 2, the same key will be derived from the GKA protocol for a soldier and the tactical units. The compromise of any one soldier will compromise the secrecy of the communication among the tactical units as well. This is also prohibitive since a soldier is conceivably under a poor communication environment.

Motivated by above scenarios, Wu *et al.* [21] introduced the notion of asymmetric group key agreement (AGKA) and proposed a concrete *one-round* AGKA protocol. Unlike regular GKA, AGKA allows the members to negotiate a common *group encryption key* while holding different *(group) decryption keys*. The group encryption key is publicly accessible and enables any sender to securely encrypt to the group members. The decryption key, which is different from the long-term private key of the user, can be used to decrypt every ciphertexts encrypted under the group encryption key.

The above AGKA protocol, and the subsequent improvements [22,23], are based on traditional public-key infrastructure (PKI). The idea of identity-based cryptosystem (IBC) proposed by Shamir [18] eliminates complicated certificate management in PKI, with the help of a trusted key generation centre (KGC) for creating the long-term identity-based secret keys for the group members. Identity-based AGKA (IBAGKA) protocols have been proposed [26,27]. Using these identity-based secret keys with AGKA, the members can securely establish a secure broadcast channel among them, without relying on PKI.

The original AGKA notion and the instantiated protocol are only secure against passive attackers who just eavesdrop the open communications. This is not sufficient against realistic attackers who may fully control the open networks

and launch more powerful active attacks such as member impersonation, communication tampering, replay of early protocol transcripts, etc. To counter this kind of attackers, an additional identity-based signature scheme is used on top of the IBAGKA protocol [26,27].

The authenticated AGKA protocol in [26,27] achieves partial forward secrecy. That is, if only one or some specific group members' long-term keys are compromised, the secrets exchanged before the compromise stay unknown to the attacker. However, if all the group members' long-term keys are leaked, then the previously established secrets will be exposed to the attacker and the protocol will no longer be secure. Obviously, since the long-term keys for the group members are generated by the KGC, the KGC can always read the secrets. This is known as the *key escrow problem*. Further, in practice, we do not know which members might be compromised after the protocol is deployed and, in the worst case, all the members and even the KGC might be compromised. These observations motivate us to investigate authenticated AGKA protocols with stronger active security.

1.1 Our Contributions

This paper contributes to the study of authenticated AGKA in the IBC setting, in the following aspects.

We first formalize the notion of IBAAGKA without escrow. Our notion captures the typical active security properties of secrecy and known-key security [26,27] 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 decryption keys of previous sessions cannot compute subsequent group decryption keys. Furthermore, our notion also captures *escrow freeness* [7] (just like the standard perfect forward secrecy [1,2]) by allowing an attacker to corrupt the KGC. True, a KGC can always generate the long-term identity-based secret keys of any user. However, even such an attacker cannot read any secret messages exchanged before the corruption.

To motivate our design of authenticated AGKA protocol, we first propose and realize our new notion of strongly unforgeable stateful identity-based batch multi-signatures (IBBMS). Borrowing its design, we propose an IBAAGKA protocol without escrow. The protocol is shown to be secure against active attacks in our strengthened model. The proof relies on the k-Bilinear Diffie-Hellman Exponent assumption (which is widely used in recent cryptographic constructions) and the strong unforgeability of our stateful IBBMS. The protocol needs only one round to enable a group of members to establish a common encryption and their respective decryption keys. A detailed analysis shows that the complexity in computation and communication of our authenticated AGKA protocol is comparable to that of up-to-date AGKA protocols, but our protocol achieves the strongest active security in AGKA protocols, so far.

1.2 Related Work

As a fundamental primitive of secure group communications, GKA has attracted considerable attention in cryptography. The best-known among these are perhaps the works of Ingemarsson *et al.* [16], Burmester and Desmedt [5], and Steiner *et al.* [20]. These proposals require two or more rounds to obtain a secret key and an additional round for each member to confirm the established secret key. Boneh and Silverberg [4] showed that a one-round GKA protocol can be constructed if multilinear maps [15] exist. However, the key confirmation step cannot be eliminated. Further, these GKA protocols only allow secure intra-group communications.

Wu *et al.* [21] constructed a one-round AGKA protocol allowing a sender not in the group to encrypt to the members while offering short ciphertexts and efficient encryption. Unlike previous GKA protocols, an interesting property of Wu *et al.*'s AGKA protocol is that it allows key confirmation without extra communication. That is, a member just needs to locally encrypt a message using the encryption key and then decrypt the corresponding ciphertext using her secret decryption key. If the messages are equal, then she obtains the keys correctly. Their protocol requires $\mathcal{O}(1)$-size ciphertext and $\mathcal{O}(1)$ encryption operations after the group encryption key is negotiated. One may note that a trivial solution of one-round AGKA is to let each member publish a public key and withhold the respective secret key. A sender can then separately encrypt to each member and can 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. The challenge is to design one-round AGKA protocols with efficient encryption and short ciphertexts.

Subsequently, Wu *et al.* strengthened AGKA and presented contributory broadcast encryption [22] so that the sender could exclude some members from reading the transmissions. In [23], Wu *et al.* showed how to shorten the size of protocol transcripts in AGKA protocols.

To alleviate complicated certificate management of authenticated GKA in the PKI setting, identity-based authenticated GKA protocols (e.g., [8,17]) have been suggested. These protocols require two or more rounds and cannot cope with sender changes. The recent IBAAGKA protocol [26,27] is one-round and can handle sender changes efficiently. However, this protocol only achieves partial forward secrecy. Further, to guarantee the security of the protocol, an additional identity-based signature is used which makes the protocol less interesting. One may consider adding random secret value(s) [6] in the key agreement phase of conventional GKA protocols to achieve *escrow freeness* in identity-based AGKA protocols. However, it is unclear how to use this method without affecting the round efficiency of AGKA protocols.

Another notion close to IBAGKA is identity-based broadcast encryption [11] due to Delerablée. However, since the long-term key derived from a member's identity is directly used for decryption, identity-based broadcast encryption

cannot even achieve partial forward secrecy and is weaker than our protocol with escrow freeness, which implies perfect forward secrecy.

Escrow freeness is especially important in the IBC setting since the KGC is the Achilles' heel and the most vulnerable spot for an attacker to break. Many works have considered solutions to address this problem. For example, forward secrecy in identity-based (anonymous) key agreement protocols [9], anonymous ciphertext indistinguishability against KGC attack in identity-based encryption [10,25], and resilience against continual auxiliary leakage of the master secret key in (hierarchical) identity-based encryption [24].

1.3 Paper Outline

The rest of the paper is organized as follows. Section 2 defines the security for IBAAGKA protocols. A strongly unforgeable stateful IBBMS signature is proposed in Sect. 3. Section 4 proposes our IBAAGKA protocol. Section 5 compares our AGKA protocol with other two AGKA protocols. Finally, Sect. 6 gives a conclusion.

2 System Model

In this section, we formalize our IBAAGKA model without escrow.

2.1 Notations

Let $\mathbb{P}$ be a polynomial-size set of participants. At any point of time, any subset $\mathbb{U} = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\} \subseteq \mathbb{P}$ may decide to establish a confidential channel. Let $\Pi^{\pi}_{\mathcal{U}_i}$ represent instance π of participant $\mathcal{U}_i$. We will require the following notations:

- $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ is the *partner ID* of $\Pi^{\pi}_{\mathcal{U}_i}$, defined by 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. The identities in $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$ are lexicographically ordered.
- $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$ is the *session ID* of instance $\Pi^{\pi}_{\mathcal{U}_i}$. The session IDs are unique. All members taking part in a given execution of a protocol have the same session ID. *The session ID of* $\Pi^{\pi}_{\mathcal{U}_i}$ *can be instantiated by concatenating* $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$, *a time interval (e.g., date of the day) 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.*
- $\mathsf{ms}^{\pi}_{\mathcal{U}_i}$ is the concatenation of all messages sent and received by $\Pi^{\pi}_{\mathcal{U}_i}$ during its execution, where the messages are ordered by round, and within each round lexicographically by the identities of the purported senders.
- $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ is the 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 receiving; and it is *successfully terminated* 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 successfully terminated; (2)* $\mathsf{pid}^{\pi}_{\mathcal{U}_i} = \mathsf{pid}^{\pi'}_{\mathcal{U}_j}$*; and (3)* $\mathsf{sid}^{\pi}_{\mathcal{U}_i} = \mathsf{sid}^{\pi'}_{\mathcal{U}_j}$.

2.2 Security Model

Our security model for IBAAGKA protocols is defined by the following game, which is run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The adversary has full control of the network communications. This game has the following stages:

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

Probing: At this stage, $\mathcal{A}$ is allowed to make 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, the query returns *null*.
- $\mathsf{Corrupt}(\mathcal{U}_i)$: It outputs the private key of participant $\mathcal{U}_i$ and can be used to model forward secrecy.
- $\mathsf{Corrupt}(\mathrm{KGC})$: It outputs the *master-secret* and can be used to model 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.
- $\mathsf{Test}(\Pi^{\pi}_{\mathcal{U}_i})$: At some point, $\mathcal{A}$ returns two messages (m_0, m_1) and a fresh instance $\Pi^{\pi}_{\mathcal{U}_i}$ (see Definition 2). $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$, 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 queried only once and is used to model secrecy.

Following [21,26,27], we use the confidentiality of the final broadcast channel to define the secrecy of IBAAGKA protocols. That is, secrecy is defined by the indistinguishability of a message encrypted under the negotiated group encryption key from a random string in the ciphertext space.

Guess: Finally, $\mathcal{A}$ returns a bit b'. If $b' = b$, $\mathcal{A}$ wins the game. $\mathcal{A}$'s advantage 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 successfully terminated.*
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* $\Pi^{\pi}_{\mathcal{U}_i}$.
3. *Before* $\Pi^{\pi}_{\mathcal{U}_i}$ *successfully terminated, the query* $\mathsf{Corrupt}$*(KGC) has been made or the query* $\mathsf{Corrupt}$*(participant) has been made for some participants whose identities are in* $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$.

Definition 3 (Secrecy). *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.*

We stress that, in our IBAAGKA secrecy definition, escrow freeness is incorporated since the attacker is allowed to corrupt the PKG. Even if such an attacker cannot understand the secret messages exchanged among the group members. The escrow freeness naturally implies perfect forward secrecy. This strong security is important in practice as IBAAGKA protocols are assumed to be deployed in ad hoc network like scenarios. In these applications, end users are usually connected by open wireless communications and exposed to attackers. Furthermore, the centralized PKG is the single point of the system and may be compromised by attackers. Our key-escrow free secrecy guarantees that IBAAGKA protocols can be securely employed in such hostile environments.

Similarly to [21,26,27], in the above model, we only consider chosen-plaintext attacks (CPA) against IBAAGKA protocols. We note our definition is readily extended to resist chosen-ciphertext (CCA) attacks. Indeed, there are some generic approaches that convert a CPA secure encryption scheme into a CCA secure one, such as the Fujisaki-Okamoto conversion [13].

3 Building Block: Strongly Unforgeable Stateful IBBMS

Here we propose a strongly unforgeable stateful IBBMS scheme as a building block of our IBAAGKA protocol.

3.1 Definition

A stateful IBBMS allows multiple signers to sign t messages under a piece of state information in an efficient way to generate a batch multi-signature. Furthermore, the batch multi-signature can be separated into t individual multi-signatures. A stateful IBBMS scheme consists of the following five algorithms:

- BM.Setup, taking as input a security parameter ℓ, outputs a *master-secret* and a list of system parameters. For brevity, we omit the inclusion of the system parameters as part of the inputs for the other algorithms.
- BM.Extract, taking as inputs an entity's identity ID_i and the *master-secret*, outputs the entity's private key.
- Sign, taking as inputs t messages, a piece of state information $info$, a signer's identity ID_i and private key, outputs a batch signature.
- Aggregate, taking as input a collection of x batch signatures on the same t messages from x signers, under the same state information $info$, outputs a batch multi-signature.
- BM.Verify, taking as input a batch multi-signature on t messages generated by x signers, under the same state information $info$, outputs either "all valid" if the batch multi-signature is valid or an index set, which means that the multi-signatures on the messages with indices in that set are valid.

As the state information, one can use the identities of all the signers and concatenate a specification of each time interval together with a counter of the number of signatures issued by these signers in the time interval.

3.2 Security Model

This section defines the strong unforgeability of stateful IBBMS schemes. Roughly speaking, a stateful IBBMS scheme is strongly unforgeable if an adversary cannot generate a different multi-signature on a message m under any state information and x signers' identities even if he can get the signature(s) on m under the same state information and identities. The formal definition of strong unforgeability of stateful IBBMS schemes is defined using 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. Υ is passed to $\mathcal{A}$ while *master-secret* is kept secret.

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

- BM.Extract: $\mathcal{A}$ can request the private key of an entity with identity ID_i. On receiving such a query, $\mathcal{C}$ outputs the private key of this entity.
- Sign: $\mathcal{A}$ can request a batch signature on messages $(m_1, \ldots, m_{t_i})$ under an identity ID_i and a piece of state information. For simplicity, we assume the messages are lexicographically ordered. On input

$$(ID_i, info_i, m_1, \ldots, m_{t_i})$$

the challenger $\mathcal{C}$ outputs a valid batch signature on those messages. If $\mathcal{A}$ asks a batch signature query with a previously used state information but different messages as input, $\mathcal{C}$ returns *null*.

Note that, to generate the IBBMS on messages $(m_1, \ldots, m_{t_i})$ under identities $(ID_1, \ldots, ID_x)$ (lexicographically ordered) and a piece of state information, $\mathcal{C}$ just needs to simulate via repeated calls to the Sign queries and then generate an IBBMS by using the Aggregate algorithm.

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 σ^*. $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. σ^* is a valid multi-signature on message m^* under identities $(ID_1^*, \ldots, ID_x^*)$ and $info^*$.
2. None of the identities in $\{ID_1^*, \ldots, ID_x^*\}$ has been submitted during the BM.Extract queries.
3. For $ID_i^* \in \{ID_1^*, \ldots, ID_x^*\}$, the forged signature σ^* is not generated by using the batch signatures output by calling the Sign queries with

$$(ID_i^*, info^*, m_1, \ldots, m_{\mathcal{I}}, \ldots, m_t)$$

as input, where $m_{\mathcal{I}} = m^*$ and $\mathcal{I}$ defines the index of the message.

In the **Forgery** stage, $\mathcal{A}$ is only required to output a single multi-signature, but not a batch multi-signature. This is due to the property of batch multi-signatures. A batch multi-signature can be separated into t individual multi-signatures. We only require that one of them is a forgery. As a result, we require that none of the identities in $\{ID_1^*, \ldots, ID_x^*\}$ has been submitted during the BM.Extract queries. This restriction is stronger than the restriction in the security models for normal multi-signature schemes which allow an adversary to query $x-1$ private keys corresponding to the identities in $\{ID_1^*, \ldots, ID_x^*\}$. However, this level of security suffices for our higher level applications in IBAAGKA. Indeed, we will be reducing the security of our IBAAGKA to that of our IBBMS.

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.*

3.3 Strongly Unforgeable Stateful IBBMS Scheme

Before delving into the details of our construction, we would like to remark that, although our final goal is not to propose an identity-based multi-signature scheme, we borrow some of its design principles to achieve our final goal of building IBAAGKA. Hence, we do not consider the generic approach of building identity-based signatures from the certification approach of standard signatures [14].

Our scheme is built over bilinear groups. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of prime order q, and g be a generator of $\mathbb{G}_1$. An efficient map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is called a bilinear map if it satisfies the following two properties.

1. **Bilinearity:** It holds that $\hat{e}(g^\alpha, g^\beta) = \hat{e}(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \mathbb{Z}_q^*$.
2. **Non-degeneracy:** There exists $u, v \in \mathbb{G}_1$ such that $\hat{e}(u, v) \neq 1$.

Now we are ready to describe our strongly unforgeable stateful IBBMS scheme.

– BM.Setup: On input a security parameter ℓ, KGC chooses two cyclic multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$ with prime order q, such that there exists a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is generated by g; KGC chooses a random $\kappa \in \mathbb{Z}_q^*$ as the *master-secret* and sets $g_{pub} = g^\kappa$; KGC chooses cryptographic hash functions

$$H_1, H_2, H_3 : \{0,1\}^* \longrightarrow \mathbb{G}_1, H_4 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$$

Finally, KGC publishes the system parameter list

$$\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{pub}, H_1 \sim H_4)$$

– BM.Extract: This algorithm takes κ and an entity's identity $ID_i \in \{0,1\}^*$ as inputs. It generates the private key for the entity as follows:

1. Compute
$$id_{i,0} = H_1(ID_i, 0), id_{i,1} = H_1(ID_i, 1)$$
2. Output the private key
$$(s_{i,0} = id_{i,0}^{\kappa}, s_{i,1} = id_{i,1}^{\kappa})$$

– Sign: To sign t messages $(m_1, \ldots, m_t)$ under a piece of state information $info$, a signer with identity ID_i and private key $(s_{i,0}, s_{i,1})$ performs the following steps:
1. Choose random $\eta_i, \theta_i \in \mathbb{Z}_q^*$ and compute
$$r_i = g^{\eta_i}, u_i = g^{\theta_i}, v = H_2(info), \varpi_i = H_4(info, ID_i, r_i, u_i)$$
$$f_j = H_3(info, m_j), z_{i,j} = s_{i,0} s_{i,1}^{\varpi_i} v^{\theta_i} f_j^{\eta_i}, \text{ for } 1 \leq j \leq t.$$
2. Output batch signature $\sigma_i = (r_i, u_i, z_{i,1}, \ldots, z_{i,t})$.

– Aggregate: Anyone can aggregate a collection of signatures
$$\{\sigma_i = (r_i, u_i, z_{i,1}, \ldots, z_{i,t})\}_{1 \leq i \leq x}$$
on the messages $\{m_j\}_{1 \leq j \leq t}$ from x signers, under same $info$, into a batch multi-signature. In particular, the signatures can be aggregated into
$$(r_1, \ldots, r_x, u_1, \ldots, u_x, d_1, \ldots, d_t), \text{ where } d_j = \prod_{i=1}^{x} z_{i,j}.$$

– BM.Verify: To check the validity of the above batch multi-signature
$$(r_1, \ldots, r_x, u_1, \ldots, u_x, d_1, \ldots, d_t)$$
the verifier computes
$$w = \prod_{i=1}^{x} r_i, y = \prod_{i=1}^{x} u_i, v = H_2(info),$$
$$f_j = H_3(info, m_j), \Gamma_j = \hat{e}(f_j, w) \text{ for } 1 \leq j \leq t,$$
$$\varpi_i = H_4(info, ID_i, r_i, u_i) \text{ for } 1 \leq i \leq x,$$
$$\Omega = \hat{e}(\prod_{i=1}^{x} H_1(ID_i, 0) H_1(ID_i, 1)^{\varpi_i}, g_{pub}) \hat{e}(v, y).$$
For $1 \leq j \leq t$, the verifier checks
$$\hat{e}(d_j, g) \stackrel{?}{=} \Omega \Gamma_j.$$
If all the equations hold, the verifier outputs "all valid"; otherwise, it outputs an index set $\mathbb{I}$, which means that the multi-signatures with indices in that set are valid.

The security of our protocol is based on the following computational Diffie-Hellman (CDH) assumption.

CDH Assumption: In a finite cyclic group $\mathbb{G}$ with order q, the CDH assumption states that, given $g, g^{\alpha}, g^{\beta} \in \mathbb{G}$ for randomly chosen $\alpha, \beta \in \mathbb{Z}_q$, there exists no efficient algorithm to compute $g^{\alpha\beta}$.

The following result relates the security of the IBBMS primitive with the difficulty of breaking the CDH assumption.

Theorem 1. *Let H_1, H_2, H_3 and H_4 be random oracles. Suppose an adversary $\mathcal{A}$ makes at most q_{H_i} queries to H_i, for $1 \leq i \leq 3$, q_E Extract queries, q_σ Sign queries with maximal message size N, and wins the game in Sect. 3.2 with advantage ϵ' in time τ'; and the forged IBBMS is by at most x users. 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

$$\tau' + \mathcal{O}(4q_{H_1} + q_{H_2} + q_{H_3} + 5Nq_\sigma)\tau_{\mathbb{G}_1}$$

where $\tau_{\mathbb{G}_1}$ is the time to compute a scalar exponentiation in $\mathbb{G}_1$ and e is Euler's number.

The proof will be presented in the full version of this paper.

4 Identity-Based Authenticated Asymmetric Group Key Agreement Protocol

In this section, we propose our one-round IBAAGKA protocol.

- Setup: It is the same as BM.Setup, except that an additional cryptographic hash function $H_5 : \mathbb{G}_2 \longrightarrow \{0,1\}^{\iota}$ is chosen, where ι defines the bit-length of plaintexts. The system parameter list is

$$\Upsilon = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{pub}, H_1 \sim H_5, \iota)$$

- Extract: It is the same as BM.Extract.
- Agreement: Assume the group scale is n and the session ID is sid_λ. A protocol participant $\mathcal{U}_i$, whose identity is ID_i and private key is $(s_{i,0}, s_{i,1})$, performs the following steps:
  1. Choose $\eta_i, \theta_i \in \mathbb{Z}_q^*$ and compute

$$r_i = g^{\eta_i}, u_i = g^{\theta_i}$$

  2. Compute

$$v = H_2(\mathsf{sid}_\lambda), \varpi_i = H_4(\mathsf{sid}_\lambda, ID_i, r_i, u_i)$$

3. For $1 \leq j \leq n$, compute
$$f_j = H_3(\mathsf{sid}_\lambda, j)$$
4. For $1 \leq j \leq n$, compute
$$z_{i,j} = s_{i,0} s_{i,1}^{\varpi_i} v^{\theta_i} f_j^{\eta_i}$$
5. Publish
$$\sigma_i = (r_i, u_i, \{z_{i,j}\}_{j \in \{1,\ldots,n\}, j \neq i})$$

– EncKeyGen: To get the group encryption key, for $j \in \{1, 2\}, i \in \{1, \ldots, n\}$, an entity computes
$$v = H_2(\mathsf{sid}_\lambda), f_j = H_3(\mathsf{sid}_\lambda, j), \varpi_i = H_4(\mathsf{sid}_\lambda, ID_i, r_i, u_i).$$
Define $\Delta = 1$, if Eqs. (1) and (2) hold, and $\Delta = 0$ in other cases.
$$\hat{e}(z_{1,2}, g) \stackrel{?}{=} \hat{e}(H_1(ID_1, 0) H_1(ID_1, 1)^{\varpi_1}, g_{pub}) \hat{e}(v, u_1) \hat{e}(f_2, r_1) \tag{1}$$
$$\hat{e}(\prod_{i=2}^{n} z_{i,1}, g) \stackrel{?}{=} \hat{e}(\prod_{i=2}^{n} H_1(ID_i, 0) H_1(ID_i, 1)^{\varpi_i}, g_{pub}) \hat{e}(v, \prod_{i=2}^{n} u_i) \hat{e}(f_1, \prod_{i=2}^{n} r_i) \tag{2}$$
The Δ is used to check whether r_i and u_i are well formatted. If $\Delta = 1$, the entity outputs (w, Ω) as the group encryption key, where
$$w = \prod_{i=1}^{n} r_i, \Omega = \hat{e}(\prod_{i=1}^{n} H_1(ID_i, 0) H_1(ID_i, 1)^{\varpi_i}, g_{pub}) \hat{e}(v, \prod_{i=1}^{n} u_i);$$
otherwise it aborts. We note that a protocol participant does not need to test the value of Δ, since it will do a similar check in the following DecKeyGen stage.

– DecKeyGen: 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 tests
$$\hat{e}(d_i, g) \stackrel{?}{=} \Omega \cdot \Gamma_i.$$
If the equation holds, $\mathcal{U}_i$ accepts d_i as the group decryption key; otherwise, it aborts. The above test is also used by $\mathcal{U}_i$ to determine whether the encryption key is valid.

– Enc: To encrypt a plaintext m, select $\rho \in \mathbb{Z}_q^*$ and compute 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 the ciphertext $c = (c_1, c_2, c_3)$, $\mathcal{U}_i$, whose group decryption key is d_i, computes

$$m = c_3 \oplus H_5(\hat{e}(d_i, c_1)\hat{e}(f_i^{-1}, c_2)).$$

The following theorem characterizes the security of our IBAAGKA protocol. The security of our protocol relies on the k-Bilinear Diffie-Hellman Exponent (BDHE) assumption [3] which states that, in the bilinear group setting, given g, h, and $g_i = g^{\alpha^i}$ in $\mathbb{G}_1$ for $i = 1, 2, \ldots, k, k+2, \ldots, 2k$ as inputs, there exists no efficient algorithm to compute $\hat{e}(g, h)^{\alpha^{k+1}}$.

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*

$$\frac{1 - 2\epsilon'}{q_{H_5}(q_{DK} + 1)e}\epsilon$$

in time

$$T = \tau + \mathcal{O}(10q_{EK})\tau_{\hat{e}} + \mathcal{O}(q_{H_2} + q_{H_3} + 2q_C + 8q_S + 3q_{EK})\tau_{\mathbb{G}_1}$$

where ϵ' is the advantage for $\mathcal{A}$ to forge a valid IBBMS in time T, $\tau_{\hat{e}}$ is the time to compute a bilinear map, $\tau_{\mathbb{G}_1}$ is the time to compute a scalar exponentiation in $\mathbb{G}_1$ and e is Euler's number.

The proof will be presented in the full version of this paper.

5 Comparison

In this section, we compare our AGKA protocol with the unauthenticated AGKA protocol in [21] and the IBAAGKA in [26,27]. We only consider the costly operations and omit the operations that can be pre-computed.

Table 1 shows the computational overhead of three protocols in the last five stages, where $\tau_{\hat{e}}, \tau_{\mathbb{G}_1}, \tau_H, \tau_{\mathbb{G}_2}, \tau_{sg}$ are the times to compute a bilinear map, a scalar exponentiation in $\mathbb{G}_1$, a MapToPoint hash, a scalar exponentiation in $\mathbb{G}_2$, and the signing algorithm of an identity-based signature (IBS), respectively. Let τ_{sv} denote the verification time of an IBS. The efficiency of an AGKA protocol is mainly determined by stages Enc and Dec, since Agreement, EncKeyGen and DecKeyGen only need to be run once. Hence, for simplicity, in EncKeyGen, we only consider the computational cost for a participant to generate the group encryption key; the computational cost for a sender to generate the group encryption key is omitted. From this table, one can find that our protocol has comparable efficiency in stages Agreement, EncKeyGen and DecKeyGen as protocols in

Table 1. Computational overhead (†: it can be done in EncKeyGen or DecKeyGen)

Protocols	Agreement	EncKeyGen	DecKeyGen	Enc	Dec
AGKA in [21]	$1\tau_{\hat{e}} + n\tau_{\mathbb{G}_1}$	–	$1\tau_{\mathbb{G}_1}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_2}$
AGKA in [26,27]	$(n+1)\tau_{\mathbb{G}_1} + n\tau_H + 1\tau_{sg}$	$1\tau_{\hat{e}}$	$2\tau_{\hat{e}} + n\tau_{sv}^{\dagger}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$
Our protocol	$(n+4)\tau_{\mathbb{G}_1} + (n+1)\tau_H$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$	$2\tau_{\hat{e}}$	$2\tau_{\mathbb{G}_1} + 1\tau_{\mathbb{G}_2}$	$2\tau_{\hat{e}} + 1\tau_{\mathbb{G}_1}$

[21,26,27]. For stages Enc and Dec, our protocol is as efficient as [26,27], and it has similar efficiency as [21].

Let P_1, P_2, P_{ID}, P_m denote the binary length of an element in $\mathbb{G}_1$, $\mathbb{G}_2$, an identity, and a message, respectively. Let P_{sig} be the length of an identity-based signature. Table 2 compares our protocol with two other protocols regarding transmission cost. From this table, one may find that the transmission overhead of our protocol is slightly lower than the one in [21,26,27] for the Agreement stage, if we consider an identity of length 160 bits. Further, the length of a ciphertext in our protocol is the same as the one in [21,26,27], assuming that the plaintexts in the three protocols are of the same size.

Table 2. Transmission Overhead

Protocols	Agreement	Ciphertext Size
AGKA in [21]	$nP_1 + P_2$	$2P_1 + P_2$
AGKA in [26,27]	$nP_1 + P_{sig} + P_{ID}$	$2P_1 + P_m$
Our protocol	$(n+1)P_1 + P_{ID}$	$2P_1 + P_m$

6 Conclusion

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

Acknowledgment. This work was supported in part by the Natural Science Foundation of China under Grants 61202465, 61021004, 11061130539, 61103222, 61173154, 61370190, 61003214, 61070192 and 61272501, the National Key Basic Research Program (973 program) under grants 2012CB315905, the Beijing Natural Science Foundation through project 4132056, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China and the Open Research Fund of Beijing Key Laboratory of Trusted Computing; the European

Commission under FP7 projects "DwB" and "Inter-Trust"; the Spanish Government under projects TIN2011-27076-C03-01 and CONSOLIDER INGENIO 2010 "ARES" CSD2007-0004; the Government of Catalonia under grant SGR2009-1135; the Shanghai NSF under Grant No. 12ZR1443500, 11ZR1411200; the Shanghai Chen Guang Program (12CG24); the Science and Technology Commission of Shanghai Municipality under grant 13JC1403500; the Fundamental Research Funds for the Central Universities of China; the Open Project of Shanghai Key Laboratory of Trustworthy Computing (No. 07dz22304201101).

The fifth author is supported by the Early Career Scheme and the Early Career Award of the Research Grants Council, Hong Kong SAR (CUHK 439713), and Direct Grant (4055018) of the Chinese University of Hong Kong.

The third author is with the UNESCO Chair in Data Privacy, but the views in this paper are his own and do not commit UNESCO.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange. In: STOC 1998, pp. 419–428 (1998)
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemp. Math. **324**, 71–90 (2003)
5. Burmester, M., Desmedt, Y.G.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
6. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. J. Inf. Secur. **6**(4), 213–241 (2007)
7. Chen, L., Kudla, C.: Identity based authenticated key agreement protocols from pairings. In: IEEE CSFW 2003, pp. 219–233 (2003)
8. Choi, K.Y., Hwang, J.Y., Lee, D.-H.: Efficient ID-based group key agreement with bilinear maps. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 130–144. Springer, Heidelberg (2004)
9. Chow, S.S.M., Choo, K.-K.R.: Strongly-secure identity-based key agreement and anonymous extension. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 203–220. Springer, Heidelberg (2007)
10. Chow, S.S.M.: Removing escrow from identity-based encryption. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 256–276. Springer, Heidelberg (2009)
11. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
12. Dutta, R., Barua, R.: Provably secure constant round contributory group key agreement in dynamic setting. IEEE Trans. Inf. Theory **54**(5), 2007–2025 (2008)
13. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

14. Galindo, D., Herranz, J., Kiltz, E.: On the generic construction of identity-based signatures with additional properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006)
15. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
16. Ingemarsson, I., Tang, D.T., Wong, C.K.: A conference key distribution system. IEEE Trans. Inf. Theory **28**(5), 714–720 (1982)
17. Reddy, K.C., Nalla, D.: Identity based authenticated group key agreement protocol. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 215–233. Springer, Heidelberg (2002)
18. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
19. Snoeyink, J., Suri, S., Varghese, G.: A lower bound for multicast key distribution. In: IEEE INFOCOM 2001, pp. 422–431 (2001)
20. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Trans. Parallel Distrib. Syst. **11**(8), 769–780 (2000)
21. Wu, Q., Mu, Y., Susilo, W., Qin, B., Domingo-Ferrer, J.: Asymmetric group key agreement. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 153–170. Springer, Heidelberg (2009)
22. Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J., Farràs, O.: Bridging broadcast encryption and group key agreement. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 143–160. Springer, Heidelberg (2011)
23. Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J., Manjón, J.A.: Fast transmission to remote cooperative groups: a new key management paradigm. IEEE/ACM Trans. Netw. **21**(2), 621–633 (2013)
24. Yuen, T.H., Chow, S.S.M., Zhang, Y., Yiu, S.M.: Identity-based encryption resilient to continual auxiliary leakage. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 117–134. Springer, Heidelberg (2012)
25. Yuen, T.H., Zhang, C., Chow, S.S.M., Liu, J.K.: Towards anonymous ciphertext indistinguishability with identity leakage. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 139–153. Springer, Heidelberg (2013)
26. Zhang, L., Wu, Q., Qin, B., Domingo-Ferrer, J.: Identity-based authenticated asymmetric group key agreement protocol. In: Thai, M.T., Sahni, S. (eds.) COCOON 2010. LNCS, vol. 6196, pp. 510–519. Springer, Heidelberg (2010)
27. Zhang, L., Wu, Q., Qin, B., Domingo-Ferrer, J.: Provably secure one-round identity-based authenticated asymmetric group key agreement protocol. Inf. Sci. **181**(19), 4318–4329 (2011)