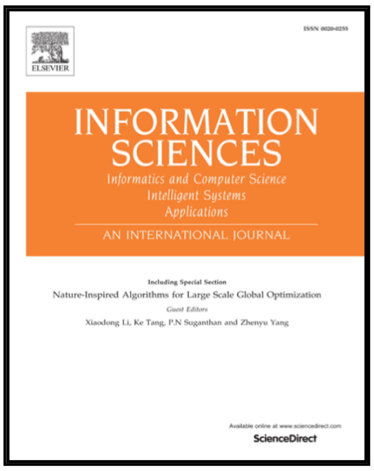

Sender Dynamic, Non-Repudiable, Privacy-Preserving and Strong Secure Group Communication Protocol

Jiangtao Li, Lei Zhang

PII: S0020-0255(17)30148-2
DOI: 10.1016/j.ins.2017.06.003
Reference: INS 12925

To appear in: *Information Sciences*

Received date: 11 January 2017
Revised date: 26 April 2017
Accepted date: 5 June 2017

Sender Dynamic, Non-Repudiable, Privacy-Preserving and Strong Secure Group Communication Protocol

Jiangtao Li, Lei Zhang*

Shanghai Key Laboratory of Trustworthy Computing, School of Computer Science and Software Engineering, East China Normal University, China

Abstract

Establishing a secure communication channel among a group of individuals is a critical issue in group oriented applications. In implementation, besides the basic confidentiality requirement, we also expect the group communications to be sender dynamic, forward secure, non-repudiable and privacy preserving. Traditional approaches, i.e., group key agreement and broadcast encryption cannot achieve the above requirements simultaneously and efficiently. In this paper, we propose our solution to secure group communications using a new security tool called identity based asymmetric group key agreement protocol with sender non-reputation and privacy. The new notion allows a group of users to negotiate a common group encryption key and their respective group decryption keys. After that, users inside/outside the group who have the knowledge of the group encryption key are able to send non-repudiable and privacy-preserving messages to the group confidentially. Further, even all the users' private keys are later corrupted, the messages broadcasted previously remain secure. Following this definition, we propose a concrete protocol which is proven to be secure against chosen-ciphertext attacks (CCA) directly. We also show the potential applications of our new protocol in instant messaging applications, such as Whatsapp, Wechat, Messenger.

Keywords: group communications, security, non-repudiation, privacy preserving, asymmetric group key agreement

*Corresponding author. Email: leizhang@sei.ecnu.edu.cn.

1. Introduction

Group oriented applications such as interactive group games, IP television and group chat, usually require secure channels to securely deliver data from an entity to a group of receivers through open networks. In order to realize such a secure group communication channel, a group key is usually shared by the group users. Only the group users who have the shared group key may decrypt the ciphertexts generated by other users. Any outsider cannot violate the confidentiality of the transmitted messages.

Besides the confidentiality requirement, other security properties, i.e., sender non-repudiation, sender privacy and forward secrecy, are also important concerns in group communications. Sender non-repudiation ensures that the messages are received unmodified, authenticated, and, a sender cannot deny the transmission of a message. Sender privacy protects the identity information of senders. It requires that an outsider cannot even distinguish whether two ciphertexts are from the same sender. In group communications, it's likely that the long-term private key of a user can be compromised by an attacker. Forward secrecy guarantees that if some group users' long-term private keys are compromised, the confidentiality of past conversations still holds. Establishing a secure group communication channel fulfilling the above security properties is a challenging issue.

Two widely used approaches for group key distribution are Group Key Agreement (GKA) [16] and Broadcast Encryption (BE) [11]. GKA enables a group of users to interactive over open networks to establish a common secret group key. With the group key, the group users may communicate with each other confidentially and anonymously. However, conventional GKA protocols suffer from several weaknesses. Firstly, when an outsider wants to send a secret message to the group users, the sender has to run a GKA protocol to negation a new group key with the current group users. This is inefficient, since the sender may change frequently. Secondly, most of the conventional GKA protocols [36, 17] require at least two rounds to establish a secret group key. If the group users are in different time zones, it seems hard for them to be online currently.

BE, which allows a sender to encrypt messages to any selected users, may eliminate the weaknesses in conventional GKA protocols. It can be classified into two categories: symmetric BE and public key BE, according to whether the encryption key can be public or not. However, the former heavily relies on a trusted dealer to generate and distribute secret group keys for the users which is essentially the key problem in secure group communications that needs to be solved. Further, it is hard to find a fully trusted third party to act as a trusted dealer in the real world. As to public key BE, existing public key BE schemes [7, 3, 14, 19] explicitly call for the identities of all the receivers as inputs for decryption. This implies the ciphertext size increases with the number of group users grows. Moreover, to achieve forward secrecy, BE schemes generally require each user to update his secret key and erase the old key at regular intervals [34]. This is inefficient and does not protect the secrecy of the ciphertexts generated in the time interval corresponding to the corrupted secret key.

Observing the weaknesses of GKA and BE, a novel approach referred to as asymmetric group key agreement (AGKA) [30, 42, 43, 31] is introduced. AGKA allows a group of users to negotiate a common group encryption key and respective group decryption key of each user. AGKA integrates the desired properties of both conventional GKA and public key BE. Compared with conventional GKA, AGKA enables any sender (who has the knowledge of the public group encryption key) to send encrypted messages to the group. Further, generally, AGKA protocols only need one round interactive to establish the group encryption/decryption keys. Hence, the group users needn't to be online at the same time, since each group user only needs to send a message and then he may leave. When compared with BE, AGKA is able to achieve forward secrecy without updating users' long-term private keys. Further, it may achieve constant-size ciphertext (i.e., the length of a ciphertext is independent of the number of users), since only the group decryption key is required when decrypting.

1.1. Related Work

As popular approaches to solve the key distribution problem, conventional GKA and BE have drawn great attention.

The first input to key agreement was the Diffie-Hellman protocol [8] which is a one round protocol. Later Joux proposed a one round key agreement for three parties [18]. However, if the number of protocol participants is greater than three, there is no efficient one round solution for conventional GKA. The best known proposals [6, 27, 9] require at least two rounds to establish a secret group key. Though one round GKA protocols can be achieved by using latest cryptography primitives, i.e., multilinear maps [12, 4] and indistinguishability obfuscation [13, 5], both of them are heavy to compute and hence not suitable for most applications. Further, before using the established group key, a key confirmation step is usually required. The existing conventional key agreement protocols call for one more round. We also note that, since all the group users hold the same group key, conventional GKA protocols also suffer from the challenge of sender dynamics.

Fiat and Naor [11], introduced the concept of BE and proposed a symmetric BE scheme. However, a symmetric BE scheme commonly calls for a trusted dealer to distribute the group key(s) to the group users [11, 29, 15]. The dealer always has the knowledge of the group key(s), which results in the key escrow problem. Further, if the group users are distributed over open networks, they essentially suffer from the problem as that of the secure group communications. The public key BE was first studied by Naor and Pinkas [24]. However, most public key BE schemes result in long ciphertexts (i.e., the ciphertext size increases linearly with the total number of the group users [46]). Though several schemes [2, 14, 19, 22] achieve constant ciphertext size, to decrypt a ciphertext, the description of the receiver set is required for a receiver. Therefore, the message needs to be transmitted still grows linearly. Forward secrecy is also a challenging issue in BE schemes. A traditional method is to adopt the concept of time intervals [34]. In each time interval, a fresh secret is generated for a group user. With this method, though an attacker cannot decrypt the ciphertexts before the corruption, the ciphertexts generated in the time interval corresponding to the corrupted secret key will be exposed. Further, those schemes usually result in long ciphertext size and/or huge computation overhead. The BE schemes [21, 5] also cope with the privacy issues. However, those schemes [21, 5] incur large ciphertext size or heavy computation overhead.

The notion of AGKA was first introduced by Wu *et al.* [30]. AGKA protocols solve the problems of sender dynamics, constant-size ciphertext and round optimization. Further, it also enables key confirmation without additional round. A group user just needs to choose a random message and encrypts it under the derived group encryption key and then decrypts the corresponding ciphertext using his group decryption key. If the output is equal to the chosen message, then the correctness of the agreed group encryption/decryption keys is confirmed. The basic AGKA protocol in [30] is only secure against passive attackers who just eavesdrop the open communications. Later, Zhang *et al.* proposed authenticated asymmetric group key agreement (AAGKA) protocols [42, 43] secure against active attackers in identity-based public key cryptosystem (ID-PKC). However, in ID-PKC, a third party called key generation centre (KGC) who is employed to issue private keys for the users in the system may always violate the secrecy of the ciphertexts. To overcome the above weakness, Zhang *et al.* proposed a novel identity-based AAGKA protocol [38] which is secure against a malicious KGC. Besides, this protocol also captures forward secrecy and known-key security (see Section 3.2). Later this protocol is extended in [39]. We note that another solution to overcome the weakness in [42, 43] is to use certificateless public key cryptosystem (CL-PKC) which can be viewed as a variation of ID-PKC. However existing AAGKA solutions in CL-PKC [28, 23, 41] do not capture known-key security or do not provide formal security analysis or only achieve partial forward secrecy (see Section 3.2). In [40], a concrete conversation from certificateless AGKA to session key escrow-free ID based AGKA is proposed. However, sender authentication and privacy issues are not considered. Recently, the problem of affiliation-hiding are studied in [32, 33]. However, the protocols in [32, 33] require a trusted/semi-trusted group authority to manage the membership.

1.2. Our Contribution

AGKA provides a new tool for secure group communications. Existing AGKA protocols may capture several desirable attributes: sender dynamics, round optimization, constant-size ciphertexts, secrecy, known-key security and forward secrecy. However, some security properties, including sender non-repudiation and sender privacy, are also desired attributes that an AGKA protocol should satisfy. Besides, existing AGKA protocols use general transformation methods (such as [10]) to achieve chosen-ciphertext attacks (CCA) secrecy. These generic constructions are usually inefficient since they usually result in long ciphertexts and remarkable computation overheads. This paper focus on AGKA protocol in ID-PKC setting which enables sender non-repudiation, sender privacy, and CCA secrecy without using generic transformation methods. The contributions of this paper are threefold.

Firstly, we formalize the notion of authenticated asymmetric group key agreement with sender non-repudiation and privacy (AAGKAwSNP) in ID-PKC setting. Similar to traditional AGKA, this notion allows a group of users to negotiate a public group encryption key and their respective group decryption keys. Knowing the group encryption key, any entity is able to deliver confidential messages to group users. However, unlike traditional AGKA which does not consider sender non-repudiation and privacy, our notion enables those security properties as well. We note that, sender non-repudiation and privacy are important. The former prevents a sender from denying the transmission of a message. The latter ensures that an outsider learns nothing about the sender identities from the transmitted ciphertexts. Traditional AGKA does not consider these properties. If a malicious sender distributes a malicious message (e.g. a virus containing software), he may later deny his misbehavior if sender non-repudiation is not considered. Besides, the sender's identity may also under threaten, which is a severe violation of the sender's privacy [21].

Secondly, we formally define the security properties, i.e., secrecy, known-key security, forward secrecy, sender non-repudiation and sender privacy that an identity based AAGKAwSNP protocol should satisfy. Secrecy denotes that even an adversary obtains the decryptions of ciphertexts of his choice, he can not decrypt a new ciphertext. In fact, our definition captures chosen-ciphertext attacks (CCA) which is stronger than chosen-plaintext attacks (CPA). We note that a protocol secure against CCA is capable of defending against active attackers who may modify messages in transit, while a CPA secure protocol cannot resistant this attack. Known-key security guarantees that if the group decryption keys of previous sessions are leaked, an attacker cannot compute subsequent group decryption keys. Forward secrecy guarantees that an attacker cannot break the secrecy of previous protocol runs even if the attacker obtains some or all users' long-term private keys. We note that our protocol achieves the strongest forward secrecy, i.e., key escrow freeness which guarantees that even the attacker corrupts the KGC who knows the long-term private keys of all the users in the system cannot break the secrecy of previous protocol runs. Sender non-repudiation ensures that a sender cannot deny his transmission of a message. Further, it also implies that the message received by a receiver is unaltered and the sender is authenticated. It requires an attacker cannot output a forgery even if the attacker is allowed to obtain signatures on many other messages of his choice. Further, sender privacy ensures that any outsider cannot even distinguish whether two ciphertexts are from the same sender.

Finally, we propose an identity based AAGKAwSNP protocol as a novel method for secure group communications. In our protocol, the public key of a user is just his identity. The private key of a user is generated and distributed by the KGC based on the identity of the user. With the private keys, a group of users can then securely establish a group encryption key and their respective group decryption keys. After that, any user may send non-repudiable and privacy-preserving messages to the group confidentially. Formal security analysis shows that our protocol achieves CCA secrecy, known-key security, key escrow freeness, sender non-repudiation and sender privacy under the k-bilinear Diffie-Hellman exponent (k-BDHE) and computational Diffie-Hellman (CDH) assumptions. We note that our protocol obtains direct CCA secrecy which does not expand the ciphertext. Hence, our construction is preferable to the existing ones based on generic transformation methods (e.g., the Fujisaki-Okamoto conversion [10]) which usually incur long ciphertext[1]. Besides, we also implement several experiments to evaluate the efficiency of our protocol. Simulation results show that both the computation and communication complexities of our protocol are comparable to those of the up-to-date AGKA protocols.

1.3. Potential Applications

Instant messaging (IM) applications (such as Whatsapp, Wechat, Messenger) are currently the most popular tools to communicate on the Internet. They could one day become major advertising platforms like Facebook and Google. A potential application of our protocol is in those IM applications. In those IM applications, when a group of users lunch a group chat, it is expected that only the group users could access the messages sent to the group. Even the IM service provider cannot eavesdrop the group communications. However, sometimes an advertiser or the IM service provider may want to send some messages (such as advertisements, service maintenance notifications, warnings of dangerous) to some specific groups (without joining the group). Obviously, it's better if the sender is authenticated and non-repudiable, so that the group users may identify the identity of the sender and investigate the responsibility of the sender if illegal message is found. Further, it is also preferable that if the IM service provider may support strong security guarantee i.e., sender privacy, known-key security, key escrow freeness and direct CCA secrecy. Using existing tools, the above problems can not be well solved simultaneously. Our protocol fits above scenario well. We note that with the method discussed in Section 4.1, our protocol may also enable user join and leave. Besides, our protocol may also be used to construct novel privacy preserving cloud data integrity checking protocol [37] which enables a cloud server to prove the integrity of a user's data to a verifier without violating the privacy of the user's data [20]. Other applications of our protocol may include group oriented applications such as interactive group games, IP television as well as secure group communications in ad-hoc networks such as vehicular ad-hoc networks (VANETs) [44].

1.4. Paper Organization

The rest of this paper is organized as follows. In Section 2, we state the background. Section 3 defines the security definitions of identity based

[1] According to [43], if Fujisaki-Okamoto conversion is applied to convert a CPA secure encryption scheme into a CCA secure one, the message expansion is one message length.

AAGKAwSNP protocol. Section 4 presents our identity based AAGKAwSNP protocol and its security results. The performance of our protocol and detailed proofs of the security results are presented in Section 5 and Section 6 respectively. Finally, Section 7 concludes the paper.

2. Background

2.1. System Architecture

Fig. 1 illustrates the system architecture, which consists of a KGC, and a group of users. The KGC is a trusted authority which is employed to generate long-term private keys for the users in the system. A group of users may run our identity based AAGKAwSNP to establish a group encryption key and their respective secret decryption keys. After that, any user inside/outside the group is able to send secret messages to the group, while only the group users who have the group decryption keys may access to the secret messages.

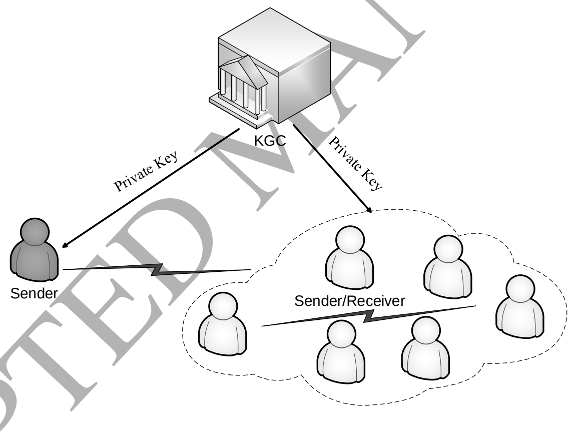

Figure 1: System architecture

2.2. Problem Statement

Suppose a set of users $\mathbb{S}$ want to establish a secure group communication channel. The problem is how to establish the channel among these users with the following constraints:

1. Both the users inside and outside of the set $\mathbb{S}$ may send confidential messages to the users in $\mathbb{S}$.
2. The source of the received messages should be authenticated and a sender cannot deny the transmission of a message.

3. The sender identity corresponding to a message should be kept secret to the ones outside of $\mathbb{S}$.

2.3. Syntax of Identity Based AAGKAwSNP

An identity based AAGKAwSNP protocol allows a group of users to securely establish a group encryption key and their respective group decryption keys. After that, any user may send non-repudiable messages to the group confidentially. Formally, an identity based AAGKAwSNP protocol consists of following algorithms:

- Setup: This algorithm is run by the KGC. It takes as input a security parameter ℓ and outputs the system global parameters and the *master-secret*.

- Extract: This algorithm is used for the KGC to generate a user's long-term private key. On input a user's identity and the *master-secret*, it outputs the user's long-term private key.

- Agreement: This is an interactive protocol that runs among the group users. Each group user generates a message and distributes the message to the intended users over public channel. This message will be used for a user to generate his group decryption key and/or group encryption key.

- Enc.KeyGen: This algorithm is used to generate a user's group encryption key. With a collection of the messages distributed by the group users, users inside/outside of the group are able to compute the common group encryption key.

- Dec.KeyGen: This algorithm is used to generate the group decryption keys. With a collection of the messages distributed by the group users, each group user is able to compute his individual group decryption key.

- Sign/Encrypt: A sender who knows the public group encryption key of a group may send a signcrypted (signed and encrypted) message to the group users using this algorithm. It takes as input a message, the sender's long-term private key and the group encryption key, and outputs a signcrypted message.

- Decrypt/Verify: This algorithm is run by a group user to decrypt and verify a signcrypted message. On input a signcrypted message and a group decryption key, it outputs a sender identity and a valid message-signature pair, or, $\perp$.

2.4. Bilinear Map and Assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order q, and P be a generater of $\mathbb{G}_1$. An efficient map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is called bilinear map if it satisfies (1) Bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $a, b \in \mathbb{Z}_q^*$, $P, Q \in \mathbb{G}_1$; (2) Non-degeneracy: There exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$.

The security of our identity based AAGKAwSNP protocol is based on the k-Bilinear Diffie-Hellman Exponent (k-BDHE) assumption and the Computational Diffie-Hellman (CDH) assumption. The k-BDHE assumption [1] states that, given P, T and $\{P_i = \alpha^i P\}_{i \in \{1,2,\ldots,k,k+2,\ldots,2k\}}$ in $\mathbb{G}_1$, it is hard to compute $\hat{e}(P, T)^{\alpha^{k+1}}$. The CDH assumption states that, given $P, aP, bP \in \mathbb{G}$, it is hard to compute abP.

2.5. Stateful Identity Based Batch Multi-Signature

The idea of identity based cryptosystem (IBC) was first introduced by Shamir [26]. Different from traditional cryptosystem, in IBC, the public key of a user is just his identifying information [35]. For instance, a user can choose his email address, domain name, or a physical IP address as his public key. No digital certificate is needed to prove the ownership of a public key in IBC. In such a system, a trusted third party named KGC is employed to initialize the system global parameters and *master-secret*. All the users have to register to the KGC and obtain their private keys issued by the KGC based on their identities.

A stateful identity based batch multi-signature (IBBMS) scheme allows multiple users to sign a batch of messages under a piece of state information in an efficient way to generate a batch multi-signature. A stateful IBBMS scheme consists of five algorithms: Setup, Extract, Sign, Aggregate and Verify. The first algorithm allows the KGC to generate the *master-secret* and the system wide parameters. On input the *master-secret* and a user's identity, the KGC may run the second algorithm to generate the private key of the user. With the private key, the user may generate a batch signature on multiple messages under a piece of state information using the third algorithm. The batch signatures on the same multiple messages under the same state information can be aggregated into a stateful IBBMS using the forth algorithm. The last algorithm is used to verify the validity of the stateful IBBMS.

In this paper, we will use the stateful IBBMS scheme in [38] as a building block of our protocol. The scheme consists of following five algorithms:

- Setup: This algorithm is run by the KGC. It takes as input a security parameter ℓ, and outputs the system global parameters and the *master-secret*. Given a security parameter ℓ, the algorithm works as follows:

1. Choose two cyclic group $\mathbb{G}_1, \mathbb{G}_2$ with prime order q, and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, choose a generater $P \in \mathbb{G}_1$.
2. Pick a random $s \in \mathbb{Z}_q^*$ as the *master-secret* and set $P_{pub} = sP$.
3. Choose four hash functions $H_1 : \{0,1\}^{l_0+l_2} \to \mathbb{G}_1, H_2 : \{0,1\}^{l_3} \to \mathbb{G}_1, H_3 : \{0,1\}^{l_3+l_4} \to \mathbb{G}_1,\ H_4 : \{0,1\}^{l_3+2l_1+l_0} \to \mathbb{Z}_q^*$, where l_0, l_1, l_2, l_3, l_4 are the bit-lengths required to represent an identity, an element in $\mathbb{G}_1$, an element in $\mathbb{G}_2$, a state information and a piece of plaintext message respectively.
4. Publish the system parameter list $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_4)$.

- Extract: This algorithm is used for the KGC to generate a user's long-term private key. It takes as input an entity's identity and the *master-secret*, outputs the private key of the entity. For a given identity id_i, the algorithm computes $Q_{i,0} = H_1(id_i, 0), Q_{i,1} = H_1(id_i, 1)$ and outputs the private key $(S_{i,0} = sQ_{i,0}, S_{i,1} = sQ_{i,1})$.

- Sign: This algorithm is run by a signer. It takes as input a piece of state information, multiple messages, a signer's identity and private key, outputs a stateful batch signature on these messages. Let the signer's identity be id_i and the corresponding private key be $(S_{i,0}, S_{i,1})$, the messages be $(m_1, \ldots, m_t)$ and the state information be $info$. The signer chooses random $r_i, z_i \in \mathbb{Z}_q^*$ and computes $R_i = r_iP,\ Z_i = z_iP,\ V = H_2(info), f_i = H_4(info, id_i, R_i, Z_i)$, and for $1 \leq j \leq t$, computes $W_j = H_3(info, m_j),\ X_{i,j} = S_{i,0} + f_iS_{i,1} + u_iV + r_iW_j$. Then the signer outputs the batch signature $\mathbb{X}_i = (R_i, Z_i, X_{i,1}, \ldots, X_{i,t})$.

- Aggregate: Anyone may run this algorithm. It aggregates multiple stateful batch signatures from different signers on the same messages and state information into a stateful batch multi-signature. On input a collection of batch signatures $\mathbb{X}_i$ from n signers on the messages $(m_1, \ldots, m_t)$ under the same $info$, it generates a stateful IBBMS $(R_1, \ldots, R_n, Z_1, \ldots, Z_n, D_1, \ldots, D_t)$, where $D_j = \sum_{i=1}^n X_{i,j}$.

- Verify: This algorithm is run by a verifier. It takes as input a stateful IBBMS, outputs either $\top$ if the signature is valid, or a set which contains the indices of the valid signatures on the corresponding messages. Given the above stateful IBBMS, the verifier computes $V =$

$H_2(info), \lambda = \hat{e}(\sum_{i=1}^{n}(Q_{i,0} + f_i Q_{i,1}), P_{pub}), W_j = H_3(info, m_j)$, where $f_i = H_4(info, id_i, R_i, Z_i), 1 \leq i \leq n, 1 \leq j \leq t$. If the equation

$$\hat{e}(D_j, P) \overset{?}{=} \lambda \hat{e}(V, \sum_{i=1}^{n} Z_i)\hat{e}(W_j, \sum_{i=1}^{n} R_i)$$

holds for all $1 \leq j \leq t$, it outputs $\top$; otherwise, it outputs a set $\mathbb{T}$ which contains the indices of valid signatures.

We note the above stateful IBBMS scheme was proven to be strongly unforgeable [38] which implies that an attacker cannot even produce a new multi-signature on a previously signed message under the same state information.

3. Security Definitions

In this section, we first define several notions, then we formally define the security attributes that an identity based AAGKAwSNP protocol should satisfy.

3.1. Notions and Definitions

Assume a set of users $\mathbb{S} = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ decide to establish a confidential channel among them. We denote instance π of participant $\mathcal{U}_i$ as $\Pi_{\mathcal{U}_i}^{\pi}$. Each instance $\Pi_{\mathcal{U}_i}^{\pi}$ is associated with following variables:

- $\mathsf{pid}_{\mathcal{U}_i}^{\pi}$ is the partner ID of $\Pi_{\mathcal{U}_i}^{\pi}$. It is a set which contains all the identities of the protocol participants, including $\mathcal{U}_i$ itself.
- $\mathsf{sid}_{\mathcal{U}_i}^{\pi}$ is the unique session ID of $\Pi_{\mathcal{U}_i}^{\pi}$. One way to instance such a session ID is to use the $\mathsf{pid}_{\mathcal{U}_i}^{\pi}$ concatenating a time interval and a counter of the session number executed by the participants who have the same partner ID $\mathsf{pid}_{\mathcal{U}_i}^{\pi}$ in that time interval.
- $\mathsf{ek}_{\mathcal{U}_i}^{\pi}$ denotes the group encryption key held by $\Pi_{\mathcal{U}_i}^{\pi}$.
- $\mathsf{dk}_{\mathcal{U}_i}^{\pi}$ denotes the group decryption key held by $\Pi_{\mathcal{U}_i}^{\pi}$.
- $\mathsf{ms}_{\mathcal{U}_i}^{\pi}$ is a set which contains all the messages sent and received by $\Pi_{\mathcal{U}_i}^{\pi}$ in a session.

Definition 1 (Successful termination)**.** *We say $\Pi_{\mathcal{U}_i}^{\pi}$ has successfully terminated if $\Pi_{\mathcal{U}_i}^{\pi}$ possesses $\{\mathsf{ek}_{\mathcal{U}_i}^{\pi}(\neq null), \mathsf{dk}_{\mathcal{U}_i}^{\pi}(\neq null), \mathsf{ms}_{\mathcal{U}_i}^{\pi}, \mathsf{pid}_{\mathcal{U}_i}^{\pi}, \mathsf{sid}_{\mathcal{U}_i}^{\pi}\}$.*

Definition 2 (Partnering). *Two instances $\Pi^{\pi}_{\mathcal{U}_i}$ and $\Pi^{\pi'}_{\mathcal{U}_j}$ are partnered if and only if : (1)$\Pi^{\pi}_{\mathcal{U}_i}$ and $\Pi^{\pi'}_{\mathcal{U}_j}$ are successfully terminated; (2) $\mathsf{sid}^{\pi}_{\mathcal{U}_i}$=$\mathsf{sid}^{\pi'}_{\mathcal{U}_j}$; (3) $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$=$\mathsf{pid}^{\pi'}_{\mathcal{U}_j}$.*

We model the capacity of an active adversary $\mathcal{A}$ by allowing him to make following various of queries:

- $\mathsf{Send}(\Pi^{\pi}_{\mathcal{U}_i}, \mu)$: It outputs the reply generated by the instance $\Pi^{\pi}_{\mathcal{U}_i}$ when a message μ is sent to $\Pi^{\pi}_{\mathcal{U}_i}$. In particular, if $\mu = (\mathsf{sid}, \mathsf{pid})$, then the session ID sid and partner ID pid will be used to initiate the protocol.
- $\mathsf{Corrupt}(\mathcal{U}_i)$: It outputs the private key of $\mathcal{U}_i$.
- $\mathsf{Corrupt}$(KGC): It outputs the *master-secret.*
- $\mathsf{EK.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$: 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 $\mathsf{dk}^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{Sign/Encrypt}(\mathsf{ek}^{\pi}_{\mathcal{U}_i}, m, id_\imath)$: It outputs the signcrypted message C, i.e., the message m is first signed by the sender whose identity is $id_\imath$ and then the message m together with its signature are encrypted under the group encryption key $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$.
- $\mathsf{Decrypt/Verify}(\Pi^{\pi}_{\mathcal{U}_i}, C)$: It decrypts the signcrypted message C and outputs the plaintext m, the corresponding signature and the sender identity $id_\imath$, if the signature is a valid signature on m under $id_\imath$.

Definition 3 (Freshness). *We say an instance $\Pi^{\pi}_{\mathcal{U}_i}$ is fresh if (1) $\Pi^{\pi}_{\mathcal{U}_i}$ has successfully terminated; (2) For all instances who are partnered with $\Pi^{\pi}_{\mathcal{U}_i}$, no $\mathsf{DK.Reveal}$ queries have been made on; (3) Before $\Pi^{\pi}_{\mathcal{U}_i}$ has successfully terminated, neither a $\mathsf{Corrupt}$(KGC) query nor a $\mathsf{Corrupt}(\mathcal{U}_j)$ query has been made, where $\mathcal{U}_j$ is a user whose identity is in $\mathsf{pid}^{\pi}_{\mathcal{U}_i}$.*

3.2. Secrecy

The secrecy of an identity based AAGKAwSNP protocol is defined using the game below. It runs 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: On input a security parameter ℓ, $\mathcal{C}$ generates the *master-secret* and initializes the system parameters denoted by Λ. Λ is passed to $\mathcal{A}$.

Phase 1: At this stage, $\mathcal{A}$ is allowed to make all the seven types of queries defined in Section 3.1.

Test: At some point, $\mathcal{A}$ returns two messages (m_0, m_1), a fresh instance $\Pi^{\pi}_{\mathcal{U}_i}$ and a sender identity $id_\imath$. $\mathcal{C}$ chooses a bit $\beta \in \{0,1\}$ at random, signcrypts m_β under $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ to produce a signcrypted message C^*, and returns C^* to $\mathcal{A}$.

Phase 2: The same as Phase 1, except that $\mathcal{A}$ is not allowed to make $\mathsf{Decrypt/Verify}(\Pi^{\pi'}_{\mathcal{U}'_i}, C^*)$ queries, where $\Pi^{\pi'}_{\mathcal{U}'_i}$ is partnered with $\Pi^{\pi}_{\mathcal{U}_i}$. In addition, the instance $\Pi^{\pi}_{\mathcal{U}_i}$ which $\mathcal{A}$ intend to attack must be kept fresh after these queries.

Guess: $\mathcal{A}$ returns a bit β'. We say that the adversary *wins* if $\beta = \beta'$. $\mathcal{A}$'s advantage is defined to be $\epsilon = |2\Pr[\beta = \beta'] - 1|$.

Definition 4. *An identity based AAGKAwSNP protocol achieves secrecy, i.e., the protocol is secure against adaptive chosen ciphertext attacks (CCA), if ϵ is negligible for any probabilistic polynomial time active adversary in the above game.*

We note that, our definition of secrecy also captures following security attributes: known key security, forward secrecy and key escrow freeness. In fact, the $\mathsf{DK.Reveal}$ queries (in phase 1 and phase 2) are used to model *known-key security* which guarantees that an attacker cannot compute subsequent group decryption keys even he knows the decryption keys of previous sessions. The $\mathsf{Corrupt}(\mathcal{U}_i)$ queries are used to model forward secrecy. We note that in the above game, $\mathcal{A}$ is allowed to obtain all the users' private keys. Hence, our game captures *perfect forward secrecy.* A weaker notion is called *partial forward secrecy* which only allows $\mathcal{A}$ to corrupt one or some specific group users' private keys. Finally, the $\mathsf{Corrupt}$(KGC) queries is used to model *key escrow freeness* which means that even if an attacker corrupts the KGC, he cannot read any secret message exchanged before the corruption. Since an attacker compromising the KGC can compute the private key of any user, key escrow freeness implies perfect forward secrecy.

3.3. Sender Non-Repudiation

An identity based AAGKAwSNP protocol achieving sender non-repudiation ensures that the sender of a signcrypted message cannot disavow his signature at a later time. It is defined via the following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The game has following stages:

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

Probe: $\mathcal{A}$ is allowed to make all but the $\mathsf{Corrupt}$(KGC) queries defined in Section 3.1.

Forge: Finally, $\mathcal{A}$ returns a signcrypted message C^* and an instance $\Pi^{\pi}_{\mathcal{U}_i}$. Assume the output of $\mathsf{Decrypt/Verify}(\Pi^{\pi}_{\mathcal{U}_i}, C^*)$ is a plaintext m, a signature σ and a sender identity $id_\imath$. We say $\mathcal{A}$ *wins* the game if (1) σ is a valid signature on m under $id_\imath$; (2) $\mathsf{Corrupt}(id_\imath)$ has never been made; (3) $\mathsf{Sign/Encrypt}(\mathsf{ek}^{\pi'}_{\mathcal{U}'_i}, m, id_\imath)$ has never been submitted, where $\mathsf{ek}^{\pi'}_{\mathcal{U}'_i}$ is any group encryption key. $\mathcal{A}$'s advantage to win the above game is defined to be $\epsilon = \Pr[\mathcal{A}\ wins]$.

Definition 5. *An identity based AAGKAwSNP protocol satisfies sender non-repudiation, if ϵ is negligible for any probabilistic polynomial time adversary in the above game.*

3.4. Sender Privacy

Sender privacy guarantees that a signcrypted message cannot help an attacker to identify the sender. Further, it guarantees that an attacker cannot even distinguish whether two signcrypted messages are from the same sender. It is defined via the following game run between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The game has following stages:

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

Phase 1: At this stage, $\mathcal{A}$ is allowed to make all the seven types of queries defined in Section 3.1.

Test: At some point, $\mathcal{A}$ returns a message m, two sender identities (id_0, id_1) and a fresh instance $\Pi^{\pi}_{\mathcal{U}_i}$. $\mathcal{C}$ chooses a bit $\beta \in \{0,1\}$ at random, signcrypts m under $\mathsf{ek}^{\pi}_{\mathcal{U}_i}$ to produce a signcrypted message C^*, and returns C^* to $\mathcal{A}$.

Phase 2: The same as Phase 1, except that $\mathcal{A}$ is not allowed to make $\mathsf{Decrypt/Verify}(\Pi^{\pi'}_{\mathcal{U}'_i}, C^*)$ queries, where $\Pi^{\pi'}_{\mathcal{U}'_i}$ is partnered with $\Pi^{\pi}_{\mathcal{U}_i}$. In addition, the instance $\Pi^{\pi}_{\mathcal{U}_i}$ which $\mathcal{A}$ intend to attack must be kept fresh after these queries.

Guess: $\mathcal{A}$ returns a bit β'. We say that the adversary $\mathcal{A}$ *wins* the game if $\beta = \beta'$. $\mathcal{A}$'s advantage is defined to be $\epsilon = |2\Pr[\beta = \beta'] - 1|$.

Definition 6. *An identity based AAGKAwSNP protocol holds sender privacy if ϵ is negligible for any probabilistic polynomial time active adversary in the above game.*

4. Our Protocol

4.1. Identity Based AAGKAwSNP Protocol

This section proposes our identity based AAGKAwSNP protocol. The protocol comes as follows:

- Setup: It is the same as the Setup in Section 2.5, except that two additional cryptographic hash functions $H_5 : \mathbb{G}_2 \to \{0,1\}^{l_0+l_1+l_4}$, and $H_6 : \{0,1\}^{l_0+l_1+l_4} \to \mathbb{Z}_q^*$ are chosen.
- $\mathsf{Extract}$: For a given identity id_i, the algorithm computes $Q_{i,0} = H_1(id_i, 0)$, $Q_{i,1} = H_1(id_i, 1), Q_{i,2} = H_1(id_i, 2)$ and outputs the private key $(S_{i,0} = sQ_{i,0}, S_{i,1} = sQ_{i,1}, S_{i,2} = sQ_{i,2})$.
- $\mathsf{Agreement}$: We assume that there are n protocol participants, each participant $\mathcal{U}_i$ possessed an identity id_i and the corresponding private key $(S_{i,0}, S_{i,1}, S_{i,2})$. We note that, in our security proof, n is related to k. It requires that n should be no more than than k. Suppose the session ID is sid_ϑ, $\mathcal{U}_i$ performs the following steps:
  1. Choose $r_i, z_i \in \mathbb{Z}_q^*$ at random, and compute $R_i = r_iP, Z_i = z_iP, V = H_2(\mathsf{sid}_\vartheta), f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i)$.
  2. For $1 \le j \le n$, compute $W_j = H_3(\mathsf{sid}_\vartheta, j), X_{i,j} = S_{i,0} + f_iS_{i,1} + z_iV + r_iW_j$.
  3. Publish $\mathbb{X}_i = (R_i, Z_i, \{X_{i,j}\}_{j\in\{1,\ldots,n\}, j\neq i})$.
- $\mathsf{Enc.KeyGen}$: To get the group encryption key, an entity computes $V = H_2(\mathsf{sid}_\vartheta)$ and $W_j = H_3(\mathsf{sid}_\vartheta, j)$ for $j \in \{1,2\}$, computes $f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i), Q_{i,0} = H_1(id_i, 0), Q_{i,1} = H_1(id_i, 1)$ for $i \in \{1, \ldots, n\}$, computes $Y = \sum_{i=2}^{n}(Q_{i,0} + f_iQ_{i,1})$ and checks the following equations:

$$\hat{e}(X_{1,2}, P) \overset{?}{=} \hat{e}(Q_{1,0} + f_1Q_{1,1}, P_{pub})\hat{e}(V, Z_1)\hat{e}(W_2, R_1) \tag{1}$$

$$\hat{e}(\sum_{i=2}^{n} X_{i,1}, P) \overset{?}{=} \hat{e}(Y, P_{pub})\hat{e}(V, \sum_{i=2}^{n} Z_i)\hat{e}(W_1, \sum_{i=2}^{n} R_i) \tag{2}$$

  If above two equations hold, set $\Upsilon = 1$; else set $\Upsilon = 0$. The Υ is used to check whether r_i and z_i are well formatted. If $\Upsilon = 1$, the entity outputs (E, ζ) as the group encryption key, where

$$E = \sum_{i=1}^{n} R_i,\ \zeta = \hat{e}(\sum_{i=1}^{n}(Q_{i,0} + f_iQ_{i,1}), P_{pub})\hat{e}(V, \sum_{i=1}^{n} Z_i);$$

otherwise it aborts. We note that only the senders who are not protocol participants need to test the value of Υ. For a protocol participant, a similar check will be done in the $\mathsf{Dec.KeyGen}$ algorithm.

- $\mathsf{Dec.KeyGen}$: To get the group decryption key, each participant $\mathcal{U}_i$ computes $E = \sum_{l=1}^{n} R_l, D_i = \sum_{l=1}^{n} X_{l,i}$ and tests $\hat{e}(D_i, P) \overset{?}{=} \zeta\hat{e}(W_i, E)$, where ζ is generated in $\mathsf{Enc.KeyGen}$. 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 group encryption key is valid.

- $\mathsf{Sign/Encrypt}$: Suppose a sender's identity is $id_\imath$ and his private key is $(S_{\imath,0}, S_{\imath,1}, S_{\imath,2})$. To signcrypt a plaintext m under the group encryption key (E, ζ), the sender performs the following steps:

  1. Choose x uniformly at random from $\mathbb{Z}_q^*$ and compute $C_1 = xP$, compute $h = H_6(C_1, m, id_\imath), F = hS_{\imath,2} + xP_{pub}$. The signature on message m under $id_\imath$ is (C_1, F).
  2. Compute $C_2 = xE, C_3 = (id_\imath, m, F) \oplus H_5(\zeta^x)$, and output the signcrypted message $C = (C_1, C_2, C_3)$.

- $\mathsf{Decrypt/Verify}$: To decrypt and verify the signcrypted message $C = (C_1, C_2, C_3)$, a user $\mathcal{U}_i$ with decryption key D_i computes

$$W_i = H_3(\mathsf{sid}_\vartheta, i), (id_\imath, m, F) = C_3 \oplus H_5(\hat{e}(D_i, C_1)\hat{e}(-W_i, C_2)),$$

  $h = H_6(C_1, m, id_\imath)$ and checks

$$\hat{e}(F, P) \overset{?}{=} \hat{e}(C_1 + hH_1(id_\imath, 2), P_{pub}).$$

  If the equation holds, it returns $(m, id_\imath, \sigma)$, where $\sigma = (C_1, F)$; else returns $\perp$.

We note that, in group communications, a user may join or leave the group for some reasons. Our protocol may also extend to support group user dynamics. To do this, we have to select one of the protocol participants as the group manager. The group manager needs to maintain a table which contains the messages broadcasted by the protocol participants in $\mathsf{Agreement}$ phase. If a user joins or leaves the group, the group manager has to update the table and/or broadcast some messages the same way as that in [39]. If a group manager leaves the group, a new manager has to be selected from the rest of the protocol participants, and the table maintained by the previous group manager has to pass to the new group manager. Hence, it is preferable to select a protocol participant who seems to stay in the group for the longest time as the group manager. After a user joins or leaves the group, each protocol participant needs to update the group encryption key and his group decryption key to the new one. We note that, the group manager could be any protocol participant and does not need to be fully trusted. This is different from a trusted dealer. Achieving dynamic AGKA without group manager is still an open problem.

4.2. Security Analysis

In this section, we show that our identity based AAGKAwSNP protocol holds all the security attributes defined in Section 3. Assume that an adversary may make at most q_{H_i} queries to H_i for $1 \leq i \leq 6$. Further, the maximum numbers of $\mathsf{Corrupt}(\mathcal{U}_i)$, $\mathsf{Dk.Reveal}(\Pi^{\pi}_{\mathcal{U}_i})$ and $\mathsf{Sign/Encrypt}(\mathsf{ek}^{\pi}_{\mathcal{U}_i}, m, id_\imath)$ queries that an adversary can make are denoted as q_c, q_{dk} and q_{se} respectively. We have following theorems which show that our identity based AAGKAwSNP protocol holds all the security attributes.

Theorem 1. *Let H_2, H_3, H_5 and H_6 be random oracles. If there is an adversary $\mathcal{A}$ that wins the game defined in Section 3.2 with probability ϵ, then there exists an algorithm $\mathcal{B}$ running in polynomial time that solves the k-BDHE problem with advantage at least $\frac{1-2\epsilon'}{q_{H_5}(q_{dk}+1)e}\epsilon$, where ϵ' is the advantage for $\mathcal{A}$ to forge a valid stateful IBBMS proposed in [38] and e is the Euler's number.*

The proof is given in Section 6.1.

Theorem 2. *Let H_1 and H_6 be random oracles. Suppose an adversary $\mathcal{A}$ wins the game defined in Section 3.3 with probability ϵ, then there exists an algorithm $\mathcal{B}$ that solves the CDH problem with probability at least $(1 - \frac{1}{q_{H_1}})^{2q_c}(1 - \frac{q_{se}q_{H_6}}{q})^2 \frac{\epsilon^2}{36q^2_{H_1}q^2_{H_6}}$.*

The proof is given in Section 6.2.

Theorem 3. *Let H_2, H_3, H_5 and H_6 be random oracles. If there is an adversary $\mathcal{A}$ that wins the game defined in Section 3.4 with probability ϵ. Then there exists an algorithm $\mathcal{B}$ running in polynomial time that solves the k-BDHE problem with advantage at least $\frac{1-2\epsilon'}{q_{H_5}(q_{dk}+1)e}\epsilon$, where ϵ' is the advantage for $\mathcal{A}$ to forge a valid stateful IBBMS proposed in [38] and e is the Euler's number.*

The proof is given in Section 6.3.

Table 1: Transmission Overhead and Security Attributes

Protocols	Agreement	Ciphertext	Authenticated	Secrecy	PFS	KEF	SNP
[30]	$nl_1 + l_2$	$2l_1 + l_2$	No	CPA[†]	No	No	No
[45]	$nl_1 + 2n(l_1 - 1)l_4$	$(n+1)l_1 + l_2$	Yes	Not Given	No	No	No
[42, 43]	$nl_1 + l_{sig} + l_0$	$2l_1 + l_4$	Yes	CPA[†]	No	No	No
[38]	$(n+1)l_1 + l_0$	$2l_1 + l_4$	Yes	CPA[†]	Yes	Yes	No
[39]	$(n+1)l_1 + l_0$	$2l_1 + l_4$	Yes	CPA[†]	Yes	Yes	No
[23][♯]	$(n+2)l_1 + l_2 + l_0$	$2l_1 + l_4$	Yes	CPA[†,♭]	No	Yes	No
[28][♯]	$(n+1)l_1 + l_0$	$2l_1 + l_4$	Yes	Not Given	No	Yes	No
[41][♯]	$(n+1)l_1 + l_0$	$2l_1 + l_4$	Yes	CPA[†]	No	Yes	No
Our protocol	$(n+1)l_1 + l_0$	$3l_1 + l_4 + l_0$	Yes	Direct CCA	Yes	Yes	Yes

[¶] PFS, KEF and SNP represent perfect forward secrecy, key escrow freeness, and, sender non-repudiation and privacy respectively.
[†] The CCA secrecy cannot be achieved directly. However, it can be achieved using generic methods which result in additionly cost.
[♯] The protocol is proposed in certificateless cryptosystem.
[♭] Known-key security is not captured.

5. Performance Evaluation

5.1. Comparison

To show the performance of the proposed protocol, we compare our protocol with the other AGKA protocols, i.e., the unauthenticated AGKA protocols in [30], the AGKA protocol in traditional cryptosystem [45], the identity based AGKA protocols in [38, 42, 43, 39] and the certificateless AGKA protocols in [41, 23, 28]. Note that, in the following, the costs of light weight operations and transmission are not taken into consideration. Besides, the comparisons with the protocols in [32] and [33] are not presented since these protocols require trusted/semi-trusted group authorities to manage the membership.

Table 1 compares our protocol with other protocols in terms of transmission overhead and security attributes. As defined in our protocol, l_0, l_1, l_2, l_3, l_4 are the bit-lengths required to represent an identity, an element in $\mathbb{G}_1$, an element in $\mathbb{G}_2$, a piece of state information and a plaintext respectively. In addition, l_{sig} denotes the bit-length of an identity based signature. As shown in Table 1, if we consider an identity of length 160 bits, the transmission overhead in the Agreement stage of our protocol is lower than the ones in [30, 42, 43, 23] and comparable with the ones in [39, 38, 41, 28]. Although the ciphertext of our protocol is slightly longer than the ones of other protocols, only our protocol achieves direct CCA secrecy, sender non-repudiation and sender privacy. Besides, our protocol also captures perfect forward secrecy and key escrow freeness which are not considered in [30, 23, 42, 43, 28, 41, 45] and [30, 42, 43, 45] respectively.

Table 2 compares the computational overhead. We use $t_{\hat{e}}$ to represent the time to compute a bilinear map, $t_{\mathbb{G}_1}$ to represent the time to compute a scalar

Table 2: Computational Overhead

Protocols	**Agreement**	**Enc.KeyGen**	**Dec.KeyGen**	**Sign/Encrypt**$^\natural$	**Decrypt/Verify**$^\natural$
[30]	$1t_{\hat{e}} + nt_{\mathbb{G}_1}$	$-^{\ddagger}$	$1t_{\mathbb{G}_1}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[45]	$4nt_{\mathbb{G}_1}$	$4nt_{\hat{e}}$	$2t_{\hat{e}}$	$(n+2)t_{\mathbb{G}_1}$	$(n+1)t_{\hat{e}}$
[42, 43]	$(n+1)t_{\mathbb{G}_1} + nt_H + 1t_{sg}$	$1t_{\hat{e}}$	$2t_{\hat{e}} + nt_{sv}{}^{\S}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[38]	$(n+4)t_{\mathbb{G}_1} + (n+1)t_H$	$2t_{\hat{e}} + (n+1)t_{\mathbb{G}_1}$	$2t_{\hat{e}}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[39]	$(n+4)t_{\mathbb{G}_1} + (n+1)t_H$	$2t_{\hat{e}} + (n+1)t_{\mathbb{G}_1}$	$2t_{\hat{e}}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[23]	$(n+1)t_{\mathbb{G}_1} + nt_H$	$(n+1)t_{\hat{e}} + nt_{\mathbb{G}_1}$	$2t_{\hat{e}}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[28]	$(n+1)t_{\mathbb{G}_1} + nt_{\hat{e}}$	$3nt_{\hat{e}} + nt_{\mathbb{G}_1}$	$3nt_{\hat{e}}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
[41]	$(n+4)t_{\mathbb{G}_1} + (n+1)t_H$	$2t_{\hat{e}} + (n+1)t_{\mathbb{G}_1}$	$2t_{\hat{e}}$	$2t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$2t_{\hat{e}} + 1t_{\mathbb{G}_1}$
Our protocol	$(n+4)t_{\mathbb{G}_1} + (n+1)t_H$	$2t_{\hat{e}} + (n+1)t_{\mathbb{G}_1}$	$2t_{\hat{e}}$	$4t_{\mathbb{G}_1} + 1t_{\mathbb{G}_2}$	$4t_{\hat{e}} + 1t_{\mathbb{G}_1} + 1t_H$

‡ Only lightweight operation is required.
§ An identity based signature is used to secure the system.
$^\natural$ Only our protocol enables Sign/Encrypt and Decrypt/Verify algorithms. In other protocols, only Encrypt and Decrypt algorithms are supported.

multiplication in $\mathbb{G}_1$, $t_{\mathbb{G}_2}$ to represent the time to compute a scalar exponentiation in $\mathbb{G}_2$, t_H to represent the time to compute a MapToPoint hash [39], t_{sg} to represent the time to generate an identity based signature and t_{sv} to represent the verification time of an identity based signature. Besides, n is the number of group participants and l_0 is the bit-length required to represent an identity. For simplicity, in the algorithm of Enc.KeyGen, we only consider the cost for a group participant to compute the group encryption key. Table 2 shows that the Agreement, Enc.KeyGen and Dec.KeyGen algorithms of our protocol have comparable efficiency with those in [39, 38, 41, 23]. Although the protocols in [42, 43] have lower computational cost in the Enc.KeyGen algorithm than that of our protocol, they require higher computational overhead in the Dec.KeyGen algorithm. One may find that our protocol requires a bit higher computational overhead in the Sign/Encrypt and Decrypt/Verify algorithms than those in other protocols. However, those algorithms have constant computational overhead and will not influence the efficiency of our protocol a lot. After all, while keeping the ciphertext size and computational overhead not greatly changed, our protocol achieves the security attributes include sender non-repudiable, send privacy and direct CCA secrecy that previous protocols do not achieve.

5.2. Experimental Analysis

The experiments were run on a PC with Intel Core i7-3770 CPU at 3.4 GHz, using the C programming language. The cryptographic operations were implemented using the Pairing-Based Cryptography (PBC) library. In our experiments, a type A curve was chosen, the security parameter ℓ was set to be 160, the protocol participants ranged from 3 to 100. The computations which could be pre-computed were not considered in the experiments.

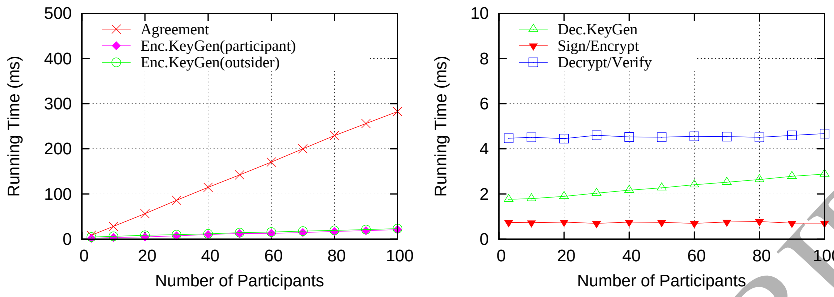

Figure 2: Average Running Time

Besides, we only evaluated the efficiency of the last 5 algorithms of our protocol. This is because the Setup algorithm only needs to be run once, while the Extract algorithm only needs to be performed when a user joins the system for the first time.

Fig. 2 shows the running time of each algorithm of our protocol. One can see that, the time cost of Agreement, Enc.KeyGen and Dec.KeyGen algorithms grow linearly as the number of participants grows. We note that the efficiency of Agreement and Enc.KeyGen algorithms will not significantly affect the efficiency of the whole protocol since the participants do not need to run them frequently. Even so, the time costs for both a group participant and an outsider to generate a group encryption key are almost the same which are less than 0.03 second when the group size is 100. Compared with the cost of Agreement algorithm, the running time of the Dec.KeyGen algorithm is much smaller which ranges from 1.76 ms to 2.88 ms when the group size ranges from 3 to 100. As the most frequently run algorithms, the time costs of Sign/Encrypt and Decrypt/Verify are constant which are only about 0.75 ms and 4.50 ms respectively.

6. Security Proofs

6.1. Proof of Theorem 1

Proof. $\mathcal{B}$ is given $(P, T, P_1, \ldots, P_k, P_{k+2}, \ldots, P_{2k})$. It simulates the challenger and interacts with $\mathcal{A}$ as follows:

Initialize: At the beginning of the game, $\mathcal{B}$ randomly chooses $s \in \mathbb{Z}_q^*$ as the *master-secret*, and sets the system parameter list

$$\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6),$$

where $P_{pub} = sP$. Λ is passed to $\mathcal{A}$. When a new session is first initiated, $\mathcal{B}$ generates a random coin $c_\vartheta \in \{0,1\}$ so that $\Pr[c_\vartheta = 1] = \delta, \Pr[c_\vartheta = 0] = 1-\delta$.

At the same time, $\mathcal{B}$ records the item $(\mathsf{sid}_\vartheta, c_\vartheta)$ where sid_ϑ is the current session ID.

Phase 1: At this stage, $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows:

H_2 queries: An initially empty list L_2 is kept by $\mathcal{B}$. On input a session ID sid_ϑ, $\mathcal{B}$ does the following:

- If there is an item $(\mathsf{sid}_\vartheta, v_\vartheta, V_\vartheta) \in L_2$, return V_ϑ as the response.
- Else, choose v_ϑ uniformly at random from $\mathbb{Z}_q^*$, set $V_\vartheta = P_1 + v_\vartheta P$, add $(\mathsf{sid}_\vartheta, v_\vartheta, V_\vartheta)$ to L_2 and return V_ϑ as the response.

H_3 queries: An initially empty list L_3 is kept by $\mathcal{B}$. On input $(\mathsf{sid}_\vartheta, j)$, $\mathcal{B}$ does the following:

- If there is an item $(\mathsf{sid}_\vartheta, j, w_j, W_j) \in L_3$, return W_j as the response.
- Else if $j \le k$, randomly choose $w_j \in \mathbb{Z}_q^*$, set $W_j = P_j + w_j P$, add $(\mathsf{sid}_\vartheta, j, w_j, W_j)$ to L_3 and return W_j as the response.
- Else, randomly choose $w_j \in \mathbb{Z}_q^*$, set $W_j = w_j P$, add $(\mathsf{sid}_\vartheta, j, w_j, W_j)$ to L_3 and return W_j as the response.

H_5 queries: An initially empty list L_5 is kept by $\mathcal{B}$. On input a message μ_i, $\mathcal{B}$ does the following:

- If there is an item $(\mu_i, \gamma_i) \in L_5$, return γ_i as the response.
- Else, choose γ_i uniformly at random from $\{0,1\}^{l_0+l_1+l_4}$, add(μ_i, γ_i) to L_5 and return γ_i as the response.

H_6 queries: An initially empty list L_6 is kept by $\mathcal{B}$. On input $(C_1, m, id_\imath)$, $\mathcal{B}$ does the following:

- If there is an item $(C_1, m, id_\imath, h) \in L_6$, return h as the response.
- Else, choose h uniformly at random from $\mathbb{Z}_q^*$, add $(C_1, m, id_\imath, h)$ to L_6 and return h as the response.

$\mathsf{Corrupt}(\mathcal{U}_i)$: Let the identity of $\mathcal{U}_i$ be id_i. On receiving the corrupt query, $\mathcal{B}$ outputs $(sH_1(id_i, 0), sH_1(id_i, 1), sH_1(id_i, 2))$.

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

$\mathsf{Send}(\Pi^\pi_{\mathcal{U}_i}, \mu)$: An initially empty list L_{sd} is kept by $\mathcal{B}$. Assume $\mathsf{sid}^\pi_{\mathcal{U}_i} = \mathsf{sid}_\vartheta$, $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{id_1, \ldots, id_n\}$ and the identity of $\mathcal{U}_i$ is id_i. In response to this query, $\mathcal{B}$ first recovers c_ϑ corresponding to sid_ϑ, recovers $(\mathsf{sid}_\vartheta, v_\vartheta, V_\vartheta)$ from L_2, for $1 \le j \le n$ recovers $(\mathsf{sid}_\vartheta, j, w_j, W_j)$ from L_3. If these items are not in the lists, it submits sid_ϑ to H_2 and $(\mathsf{sid}_\vartheta, j)$ to H_3 to get the responses. $\mathcal{B}$ computes $Q_{i,0} = H_1(id_i, 0), Q_{i,1} = H_1(id_i, 1)$, and then does the following:

- If $c_\vartheta = 0$, compute $S_{i,0} = sQ_{i,0}, S_{i,1} = sQ_{i,1}$ and do the following:
  1. Choose $r_i, z_i \in \mathbb{Z}_q^*$ and compute $R_i = r_iP, Z_i = z_iP$.
  2. Compute $f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i)$.
  3. For $1 \leq j \leq n$, compute $X_{i,j} = S_{i,0} + f_iS_{i,1} + z_iV_\vartheta + r_iW_j$.
  4. Add $(id_i, \mathsf{sid}_\vartheta, r_i, z_i, X_{i,i})$ to L_{sd} and return

     $$\mathbb{X}_i = (R_i, Z_i, \{X_{i,j}\}_{j\in\{1,\ldots,n\},j\neq i})$$

     as the response.

- Else if $c_\vartheta = 1$, choose an index $i^* \in \{1, \ldots, n\}$ uniformly at random and proceed as follows:

  – If $i \neq i^*$, choose $r_i, z_i \in \mathbb{Z}_q^*$ uniformly at random, compute $R_i = r_iP + P_{k-i+1}$, $Z_i = z_iP, f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i)$, for $1 \leq j \leq n, j \neq i$, compute $X_{i,j} = Y_{i,j} + P_{k-i+1+j}$, where $Y_{i,j} = v_iZ_i + w_iR_i + z_iP_1 + r_iP_j + s(Q_{i,0} + f_iQ_{i,1})$, return

  $$(R_i, Z_i, \{X_{i,j}\}_{j\in\{1,\ldots,n\},j\neq i})$$

  as the response and add $(id_i, \mathsf{sid}_\vartheta, r_i, z_i, \perp)$ to L_{sd}.

  – Else if $i = i^*$, choose $r_i, z_i \in \mathbb{Z}_q^*$ uniformly at random, compute $R_i = r_iP + \sum_{l=1}^{n,l\neq i} P^{-1}_{k-l+1}$, $Z_i = z_iP + P_k$, $f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i)$, for $1 \leq j \leq n, j \neq i$, compute $X_{i,j} = Y_{i,j} + \sum_{l=1}^{n,l\neq i} P^{-1}_{k-i+1+j}$ where $Y_{i,j} = v_iZ_i + w_iR_i + z_iP_1 + r_iP_j + s(Q_{i,0} + f_iQ_{i,1})$, return

  $$(R_i, Z_i, \{X_{i,j}\}_{j\in\{1,\ldots,n\},j\neq i})$$

  as the response and add $(id_i, \mathsf{sid}_\vartheta, r_i, z_i, \perp)$ to L_{sd}.

$\mathsf{EK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$: Assume $\mathsf{sid}^\pi_{\mathcal{U}_i} = \mathsf{sid}_\vartheta$, $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{id_1, \ldots, id_n\}$ and $\mathsf{ms}^\pi_{\mathcal{U}_i} = \{\mathbb{X}_1, \ldots, \mathbb{X}_n\}$, where $\mathbb{X}_l = (R_l, Z_l, \{X_{l,j}\}_{j\in\{1,\ldots,n\},j\neq l})$. If $\Pi^\pi_{\mathcal{U}_i}$ is successfully terminated and $\Upsilon = 1$, $\mathcal{B}$ recovers $V = H_2(\mathsf{sid}_\vartheta)$ from L_2, for $l \in \{1, \ldots, n\}$ computes $Q_{l,0} = H_1(id_l, 0), Q_{l,1} = H_1(id_l, 1), f_l = H_4(\mathsf{sid}_\vartheta, id_l, R_l, Z_l)$, computes

$$E = \sum_{l=1}^{n} R_l, \zeta = \hat{e}(\sum_{l=1}^{n}(Q_{l,0} + f_lQ_{l,1}), P_{pub})\hat{e}(V, \sum_{l=1}^{n} Z_l),$$

and returns (E, ζ) as the response; otherwise, it returns $\perp$.

$\mathsf{DK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$: Suppose the current session ID is sid_ϑ. In response to this query, $\mathcal{B}$ first recovers c_ϑ corresponding to sid_ϑ, and then it does the following:

- If $c_\vartheta = 1$, abort (*Event* 1).
- Else if $\Pi^\pi_{\mathcal{U}_i}$ is not successfully terminated, return $\perp$.
- Else, suppose $\mathsf{ms}^\pi_{\mathcal{U}_i} = \{\mathbb{X}_1, \ldots, \mathbb{X}_n\}$, where

$$\mathbb{X}_l = (R_l, Z_l, \{X_{l,j}\}_{j \in \{1,\ldots,n\}, j \neq l}),$$

recover the corresponding $X_{i,i}$ from L_{sd}, compute and output $D_i = \sum_{l=1}^n X_{l,i}$ as the response.

$\mathsf{Sign/Encrypt}(\mathsf{ek}^\pi_{\mathcal{U}_i}, m, id_\imath)$: If $\Pi^\pi_{\mathcal{U}_i}$ is successfully terminated, we have $\mathsf{ek}^\pi_{\mathcal{U}_i} \neq null$. Assume $\mathsf{ek}^\pi_{\mathcal{U}_i} = (E, \zeta)$. $\mathcal{B}$ chooses x uniformly at random from $\mathbb{Z}_q^*$, computes $S_{\imath,2} = sH_1(id_\imath, 2), C_1 = xP$ and recovers h corresponding to $(C_1, m, id_\imath)$ from L_6, computes $C_2 = xE, C_3 = (id_\imath, m, hS_{\imath,2} + xP_{pub}) \oplus H_5(\zeta^x)$, and returns (C_1, C_2, C_3) as the response. Else if $\Pi^\pi_{\mathcal{U}_i}$ is not successfully terminated, it returns $\perp$.

$\mathsf{Decrypt/Verify}(\Pi^\pi_{\mathcal{U}_i}, C)$: Assume $\mathsf{sid}^\pi_{\mathcal{U}_i} = \mathsf{sid}_\vartheta$ and $C = (C_1, C_2, C_3)$. In response to this query, $\mathcal{B}$ first recovers c_ϑ corresponding to sid_ϑ. If $c_\vartheta = 0$, $\mathcal{B}$ recovers the group decryption key and then decrypts the signcrypted message; else if $c_\vartheta = 1$, $\mathcal{B}$ steps through the items $(\mu_i, \gamma_i) \in L_5$ for $i \in \{1, \ldots, q_{H_5}\}$ and does the following:

1. Compute $(id_\imath, m, F) = C_3 \oplus \gamma_i$.
2. Step through the list L_6, if there is an item $(C_1^*, m^*, id_\imath^*, h^*)$ such that $m^* = m, id_\imath^* = id_\imath$, recover the corresponding C_1^* and h^* and turn to the next step; else move to the next element in L_5 and go back to step 1.
3. If $C_1^* = C_1$, turn to the next step; else move to the next element in L_5 and go back to step 1.
4. Check $\hat{e}(F, P) \overset{?}{=} \hat{e}(C_1^* + h^* H_1(id_\imath, 2), P_{pub})$. If the equation holds, return $(m, id_\imath, (C_1, F))$; else, move to the next item in L_5 and go back to step 1.

After stepping through L_5, if no message has been returned, return $\perp$.

Test: At some point, $\mathcal{A}$ chooses two messages (m_0, m_1), a fresh instance $\Pi^\pi_{\mathcal{U}_i}$ and a sender identity $id_\imath$. Assume the session ID is sid^*_ϑ, $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{id_1^*, \ldots, id_n^*\}$ and $\mathsf{ms}^\pi_{\mathcal{U}_i} = \{\mathbb{X}_1^*, \ldots, \mathbb{X}_n^*\}$, where

$$\mathbb{X}_l^* = (R_l^*, Z_l^*, \{X_{l,j}^*\}_{j \in \{1,\ldots,n\}, j \neq l}).$$

$\mathcal{B}$ does the following:

- Recover c^*_ϑ corresponding to sid^*_ϑ. If $c^*_\vartheta = 1$ and $\Upsilon = 1$ turn to the next step; otherwise, abort (*Event* 2).

- Recover $(\mathsf{sid}^*_\vartheta, v^*_\vartheta, V^*_\vartheta)$ from L_2 and $(r^*_l, z^*_l)_{l\in 1\ldots,n}$ from L_{sd}. Since the underlying stateful IBBMS is strongly unforgeable, with probability $1-2\epsilon'$, (R^*_l, Z^*_l) is well formatted, where ϵ' is the advantage for $\mathcal{A}$ to forge a valid IBBMS in [38]. If so, compute $Q^*_{l,0} = H_1(id^*_l, 0), Q^*_{l,1} = H_1(id^*_l, 1), f^*_l = H_4(\mathsf{sid}^*_\vartheta, id^*_l, r^*_l P, z^*_l P)$. Then we have $\mathsf{ek}^\pi_{\mathcal{U}_i} = (E^*, \zeta^*)$, where

$$E^* = \sum_{l=1}^{n} r^*_l P,$$

$$\zeta^* = \hat{e}(\sum_{l=1}^{n}(Q^*_{l,0} + f^*_l Q^*_{l,1}), sP)\hat{e}(P_1 + v^*_\vartheta P, \sum_{l=1}^{n} z^*_l P + P_k).$$

- Choose $\beta \in \{0,1\}$ at random, set $C^*_1 = T, C^*_2 = \sum_{l=1}^{n} r^*_l T$, compute $S_{\imath,2} = sH_1(id_\imath, 2)$ and $h = H_6(C_1, m_\beta, id_\imath)$, then randomly choose $\gamma^* \in \{0,1\}^{l_0+l_1+l_4}$, and compute $C^*_3 = (id_\imath, m_\beta, hS_{\imath,2} + sT) \oplus \gamma^*$.

- Return $C^* = (C^*_1, C^*_2, C^*_3)$.

Phase 2: The same as **Phase 1**, except that $\mathcal{A}$ is not allowed to make a $\mathsf{Decrypt/Verify}(\Pi^{\pi'}_{\mathcal{U}'_i}, C^*)$ query, where $\Pi^{\pi'}_{\mathcal{U}'_i}$ is partnered with $\Pi^{\pi}_{\mathcal{U}_i}$. In addition, the instance $\Pi^{\pi}_{\mathcal{U}_i}$ which $\mathcal{A}$ intend to attack must be kept fresh after these queries.

Response: Once $\mathcal{A}$ finishes inquiry, $\mathcal{A}$ returns his guess $\beta' \in \{0,1\}$. If $\beta' \neq \beta$, $\mathcal{B}$ aborts the game (*Event* 3); otherwise $\mathcal{B}$ randomly chooses an item (μ_i, γ_i) from L_5 and returns the value

$$\mu_i(\hat{e}(\sum_{l=1}^{n}(Q^*_{l,0} + f_l Q^*_{l,1}), sT)\hat{e}(v^*_\vartheta T, J^*)\hat{e}(P_1, \sum_{l=1}^{n} z_l T))^{-1}$$

as its guess at the solution to the k-BDHE problem, where $Q^*_{l,0} = H_1(id^*_l, 0)$, $Q^*_{l,1} = H_1(id^*_l, 1), J^* = \sum_{l=1}^{n} Z^*_l = \sum_{l=1}^{n} z^*_l P + P_k$.

It is easy to see that our game will not abort if none of *Event* 1, *Event* 2 or *Event* 3 happens. Since $\Pr[\neg \textit{Event } 1] \geq (1-\delta)^{q_{dk}}, \Pr[\neg \textit{Event } 2] \geq \delta(1-2\epsilon'), \Pr[\neg \textit{Event } 3] \geq \epsilon$, and these events are independent, the overall probability that $\mathcal{B}$ does not abort is $\Pr[\neg \textit{Event } 1 \wedge \neg \textit{Event } 2 \wedge \neg \textit{Event } 3] = \delta(1-\delta)^{q_{dk}}(1-2\epsilon')\epsilon$.

If our game does not abort, then $\mathcal{B}$ simulates a real attack environment for $\mathcal{A}$. Therefore, $\mathcal{A}$'s advantage to win the game is ϵ. Further, the probability for $\mathcal{B}$ to choose the required μ_i is $\frac{1}{q_{H_5}}$. The overall advantage for $\mathcal{B}$ to solve the k-BDHE problem is at least $\frac{1-2\epsilon'}{q_{H_5}(q_{dk}+1)e}\epsilon$.

□

6.2. Proof of Theorem 2

Proof. Let (P, aP, bP) be an instance of the CDH problem. We now show that an algorithm $\mathcal{B}$ can output the solution to the CDH problem with the help of an adversary $\mathcal{A}$. In the following, we first show how to provide $\mathcal{A}$ with a consistent view and then introduce how to output the solution (using the technique introduced by Pointcheval and Stern [25]). $\mathcal{B}$ consists of three sub-algorithms: $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$. $\mathcal{B}_1$ simulates the challenger and interacts with $\mathcal{A}$ as follows:

Initialize: At the beginning of the game, $\mathcal{B}_1$ generates the system parameter list $\Lambda = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6)$ in which $P_{pub} = bP$. Λ is passed to $\mathcal{A}$.

Probe: $\mathcal{B}_1$ responds to $\mathcal{A}$'s queries as follows:

H_1 queries: At the beginning, $\mathcal{B}_1$ keeps an initially empty list L_1 and chooses an index τ at random from $\{1, \ldots, q_{H_1}\}$. $\mathcal{B}_1$ responds to $\mathcal{A}$'s i-th query (id_i, ϱ), $\varrho \in \{0, 1, 2\}$ as follows:

- If $i = \tau$ and $\varrho = 2$, set $id_\tau = id_i$ and return aP as the response.
- Else if $i = \tau$ and $\varrho \neq 2$, abort (*Event* 4).
- Else, choose $y_{i,\varrho}$ at random from $\mathbb{Z}_q^*$, compute $Q_{i,\varrho} = y_{i,\varrho}P, S_{i,\varrho} = y_{i,\varrho}P_{pub}$, add $(id_i, Q_{i,\varrho}, S_{i,\varrho}, y_{i,\varrho}, \varrho)$ to L_1 and return $Q_{i,\varrho}$ as the response.

H_6 queries: The same as that in Section 6.1.

Corrupt$(\mathcal{U}_i)$: Assume the identity of $\mathcal{U}_i$ is id_i and the queries $H_1(id_i, 0)$, $H_1(id_i, 1)$, $H_1(id_i, 2)$ have already been submitted. $\mathcal{B}_1$ responds to the queries as follows:

- If $id_i = id_\tau$, abort (*Event* 5).
- Else, search L_1 for the items $(id_i, Q_{i,0}, S_{i,0}, y_{i,0}, 0)$, $(id_i, Q_{i,1}, S_{i,1}, y_{i,1}, 1)$ and $(id_i, Q_{i,2}, S_{i,2}, y_{i,2}, 2)$ corresponding to id_i and return $(S_{i,0}, S_{i,1}, S_{i,2})$ as the response.

Send$(\Pi^{\pi}_{\mathcal{U}_i}, \Psi)$: An initially empty list L_{sd} is kept by $\mathcal{B}_1$. Assume that $\mathcal{A}$ has already queried $H_1(id_i, 0), H_1(id_i, 1)$. Further, assume the current session ID is sid_ϑ, $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{id_1, \ldots, id_n\}$ and the identity of $\mathcal{U}_i$ is id_i. To answer this query, $\mathcal{B}_1$ first searches L_1 for the items $(id_i, Q_{i,0}, S_{i,0}, y_{i,0}, 0)$ and $(id_i, Q_{i,1}, S_{i,1}, y_{i,1}, 1)$ corresponding to id_i and then does the following:

1. Choose $r_i, z_i \in \mathbb{Z}_q^*$ uniformly at random and compute $R_i = r_iP, Z_i = z_iP$.
2. Compute $V = H_2(\mathsf{sid}_\vartheta), f_i = H_4(\mathsf{sid}_\vartheta, id_i, R_i, Z_i)$.
3. For $1 \leq j \leq n$, compute $W_j = H_3(\mathsf{sid}_\vartheta, j), X_{i,j} = S_{i,0} + f_iS_{i,1} + z_iV + r_iW_j$.
4. Output $\mathbb{X}_i = (R_i, Z_i, \{X_{i,j}\}_{j\in\{1,\ldots,n\},j\neq i})$ and add $(id_i, \mathsf{sid}_\vartheta, r_i, z_i, X_{i,i})$ to L_{sd}.

$\mathsf{EK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$: The same as that in Section 6.1.
$\mathsf{DK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$: Assume $\mathsf{ms}^\pi_{\mathcal{U}_i} = \{\mathbb{X}_1, \ldots, \mathbb{X}_n\}$, where

$$\mathbb{X}_l = (R_l, Z_l, \{X_{l,j}\}_{j\in\{1,\ldots,n\},j\neq l}).$$

In response to this query, $\mathcal{B}_1$ does the following:

- If $\Pi^\pi_{\mathcal{U}_i}$ is not successfully terminated, return $\perp$.
- Else, recover the corresponding $X_{i,i}$ from L_{sd}, compute and output $D_i = \sum_{l=1}^n X_{l,i}$.

$\mathsf{Sign/Encrypt}(\mathsf{ek}^\pi_{\mathcal{U}_i}, m, id_\imath)$: Assume that $\mathcal{A}$ has already queried $H_1(id_\imath, \varrho)$, where $\varrho \in \{0, 1, 2\}$. Suppose $\mathsf{ek}^\pi_{\mathcal{U}_i} = (E, \zeta)$, $\mathsf{sid}^\pi_{\mathcal{U}_i} = \mathsf{sid}_\vartheta$, $\mathsf{pid}^\pi_{\mathcal{U}_i} = \{id_1, \ldots, id_n\}$ and $\mathsf{ms}^\pi_{\mathcal{U}_i} = \{\mathbb{X}_1, \ldots, \mathbb{X}_n\}$, where $\mathbb{X}_l = (R_l, Z_l, \{X_{l,j}\}_{j\in\{1,\ldots,n\},j\neq l})$. In response to this query, $\mathcal{B}_1$ does the following:

- If $\Pi^\pi_{\mathcal{U}_i}$ is not successfully terminated, return $\perp$.
- Else if $id_\imath = id_\tau$, $\mathcal{B}_1$ does the following:
  1. Choose $x', h \in \mathbb{Z}_q^*$ at random and compute $C_1 = x'P - haP$. If there is an item $(C_1^*, m^*, id_\imath^*, h^*)$ in L_6 such that $C_1^* = c_1, m^* = m, id_\imath^* = id_\imath, h^* \neq h$, abort the game (*Event* 6); else, add $(C_1, m, id_\imath, h)$ to L_6.
  2. For $l \in \{1, \ldots, n\}$, $\varrho \in \{0, 1\}$, compute $f_l = H_4(\mathsf{sid}_\vartheta, id_\imath, R_l, Z_l)$ and recover $(id_\imath, Q_{l,\varrho}, S_{l,\varrho}, y_{l,\varrho})$, $(id_\imath, \mathsf{sid}_\vartheta, r_l, z_l, X_{l,l})$ from L_1, L_{se} respectively.
  3. Compute $V = H_2(\mathsf{sid}_\vartheta), C_2 = \sum_{l=1}^n r_l(x'P - haP)$, $C_3 = (id_\imath, m, x'bP) \oplus (\hat{e}(x'P - haP, P_{pub})^{\sum_{l=1}^n(y_{l,0}+f_ly_{l,1})}\hat{e}(V, x'P - haP)^{\sum_{l=1}^n z_l})$ and return (C_1, C_2, C_3).

- Else, first recover $S_{\imath,2}$ corresponding to $id_\imath$ from L_1, choose $x \in \mathbb{Z}_q^*$ at random, compute $C_1 = xP$, $h = H_6(C_1, m, id_\imath)$, $C_2 = xE, C_3 = (id_\imath, m, hS_{\imath,2} + xP_{pub}) \oplus H_5(\zeta^x)$, and then return (C_1, C_2, C_3) as the response.

$\mathsf{Decrypt/Verify}(\Pi^\pi_{\mathcal{U}_i}, C)$: In response to this query, $\mathcal{B}_1$ first computes $\mathcal{U}_i$'s decryption key $D_i = \sum_{l=1}^n X_{l,i}$ by calling $\mathsf{DK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$ and computes $W_i = H_3(\mathsf{sid}_\vartheta, i)$, $(id_\imath, m, F) = C_3 \oplus H_5(\hat{e}(D_i, C_1)\hat{e}(W_i^{-1}, C_2))$, then computes $h = H_6(C_1, m, id_\imath)$ and checks

$$\hat{e}(F, P) \stackrel{?}{=} \hat{e}(C_1 + hH_1(id_\imath, 2), P_{pub}).$$

If the equation holds, $\mathcal{B}_1$ returns $(m, id_\imath, (C_1, F))$; else returns $\perp$.

Forge: Finally, $\mathcal{A}$ returns a signcrypted message $C^* = (C_1^*, C_2^*, C_3^*)$ and an instance $\Pi^\pi_{\mathcal{U}_i}$. $\mathcal{B}_1$ computes $\mathcal{U}_i$'s decryption key $D_i = \sum_{l=1}^n X_{l,i}$ by calling $\mathsf{DK.Reveal}(\Pi^\pi_{\mathcal{U}_i})$, computes $W_i = H_3(\mathsf{sid}_\vartheta, i)$, $(id_\imath, m, F) = C_3^* \oplus H_5(\hat{e}(D_i, C_1^*)\hat{e}(-W_i, C_2^*))$, computes $h = H_6(C_1^*, m, id_\imath)$. It requires that (C_1^*, F) is a valid and nontrivial forgery; otherwise, $\mathcal{B}_1$ aborts (*Event* 7).

It is easy to see that $\mathcal{B}_1$ will not abort if none of *Event* 4, *Event* 5, *Event* 6 or *Event* 7 happens. We have to calculate $\Pr[\neg \textit{Event } 4 \wedge \neg \textit{Event } 5 \wedge \neg \textit{Event } 6 \wedge \neg \textit{Event } 7]$. It is easy to see $\Pr[\neg \textit{Event } 4] = 1/3$, $\Pr[\neg \textit{Event } 5] \geq (1 - \frac{1}{q_{H_1}})^{q_c}$, $\Pr[\neg \textit{Event } 6] \geq 1 - \frac{q_{se}q_{H_6}}{q}$, $\Pr[\neg \textit{Event } 7] \geq \frac{\epsilon}{q_{H_1}}$, where q is the order of the group $\mathbb{G}_1$. Hence, the overall probability that $\mathcal{B}_1$ does not abort is $\xi \geq (1 - \frac{1}{q_{H_1}})^{q_c}(1 - \frac{q_{se}q_{H_6}}{q})\frac{\epsilon}{3q_{H_1}}$.

Since we require that (C_1^*, F) is a valid signature, $\mathcal{A}$ has queried $H_6(C_1^*, m, id_\imath)$ at some point during the game with overwhelming probability. We say this query is a *critical query*. Let Θ be the random taps for random oracle H_6, and Ξ be the random taps for all functions of $\mathcal{B}_1$ except random oracle H_6. $\mathcal{B}_2$ chooses $j \in \{1, \ldots, q_{H_6}\}$ at random, where q_{H_6} is the times that H_6 is called in above simulation. We now split Θ into Θ_1 and Θ_2, where Θ_1 contains the random responses to queries $1, \ldots, j-1$ and Θ_2 contains the random responses to queries $j, \ldots, q_{H_6}$. $\mathcal{B}_2$ runs the former game with the same Ξ and Θ_1 but a new Θ_2, say Θ_2'. With probability $1/q_{H_6}$, the value of j we chosen is corresponding to the critical query. Further, if $\mathcal{B}_2$ does not abort, $\mathcal{A}$ will output a forgery $(C_1^{*\prime}, F')$.

Let $X = \Xi \cup \Theta_1$ and $Y = \Theta_2$. According to the following Splitting Lemma 1, with probability at least $(1 - \frac{1}{q_{H_1}})^{2q_c}(1 - \frac{q_{se}q_{H_6}}{q})^2\frac{\epsilon^2}{36q_{H_1}^2q_{H_6}^2}$, $\mathcal{A}$ outputs two signatures (C_1^*, F) and $(C_1^{*\prime}, F')$ on m under $id_\imath$ such that $C_1^* = C_1^{*\prime}, h \neq h'$ and $F \neq F'$, where h and h' are the responses to the critical queries in $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively.

Lemma 1 (Splitting Lemma [25]). *Let $A \subset X \times Y$ be such that* $\Pr[(x,y) \in A] \geq \xi$. *Define*

$$B = \{(x,y) \in X \times Y \mid Pr_{y' \in Y}[(x,y') \in A] \geq \xi/2\}.$$

We have the following

1. $\forall (x,y) \in B,\ Pr_{y' \in Y}[(x,y') \in A] \geq \xi/2.$
2. $\Pr[B|A] \geq 1/2.$

Since the two signatures should be valid, we have $\hat{e}(F,P) = \hat{e}(C_1^* + hH_1(id_\imath, 2), P_{pub})$ and $\hat{e}(F',P) = \hat{e}(C_1^* + h'H_1(id_\imath, 2), P_{pub})$. Note that P_{pub} is equal to bP. The algorithm $\mathcal{B}_3$ can compute

$$abP = (h - h')^{-1}(F - F').$$

□

6.3. Proof of Theorem 3

Proof. Suppose that $\mathcal{A}$ is an adversary who can win the game defined in Section 3.4 with probability ϵ. Given $(P, T, P_1, \ldots, P_k, P_{k+2}, \ldots, P_{2k})$, we now show that an algorithm $\mathcal{B}$ can output the solution to the k-BDHE problem in polynomial time. $\mathcal{B}$ simulates the challenger and interacts with $\mathcal{A}$ as follows:

Initialize: The same as the **Initialize** stage in Section 6.1.

Phase 1: The same as the **Phase 1** in Section 6.1.

Test: At some point, $\mathcal{A}$ outputs a message m, a fresh instance $\Pi_{\mathcal{U}_i}^{\pi}$ and two sender identities $(id_{\imath,0}, id_{\imath,1})$ which he wishes to challenge. Assume $\mathsf{sid}_{\mathcal{U}_i}^{\pi} = \mathsf{sid}_{\vartheta}^{*}, \mathsf{pid}_{\mathcal{U}_i}^{\pi} = \{id_1^*, \ldots, id_n^*\}$ and $\mathsf{ms}_{\mathcal{U}_i}^{\pi} = \{\mathbb{X}_1^*, \ldots, \mathbb{X}_n^*\}$, where $\mathbb{X}_l^* = (R_l^*, Z_l^*, \{X_{l,j}^*\}_{j \in \{1,\ldots,n\}, j \neq l})$. $\mathcal{B}$ does the following:

- Recover c_ϑ^* corresponding to sid_ϑ^*. If $c_\vartheta^* = 1$ and $\Upsilon = 1$ turn to next step; otherwise, abort.
- Recover $(\mathsf{sid}_\vartheta^*, v_\vartheta^*, V_\vartheta^*)$ from L_2 and $(r_l^*, z_l^*)_{l \in 1\ldots,n}$ from L_{sd}. Since the underlying stateful IBBMS is strongly unforgeable, with probability $1 - 2\epsilon'$, (R_l^*, Z_l^*) is well formatted, where ϵ' is the advantage for $\mathcal{A}$ to forge a valid IBBMS in [38]. We have

$$\mathsf{ek}_{\mathcal{U}_i}^{\pi} = (E^*, \zeta^*) = (\sum_{l=1}^{n} r_l^* P, \hat{e}(Y^*, sP)\hat{e}(P_1 + v_l^* P, J^*)),$$

  where $Y^* = \sum_{l=1}^{n}(Q_{l,0}^* + f_l^* Q_{l,1}^*), f_l^* = H_4(\mathsf{sid}_\vartheta^*, id_l^*, R_l^*, Z_l^*), Q_{l,0}^* = H_1(id_l^*, 0), Q_{l,1}^* = H_1(id_l^*, 1), J^* = \sum_{l=1}^{n} Z_l^* = \sum_{l=1}^{n} z_l^* P + P_k$.

- Choose $\beta \in \{0,1\}$ at random, set $C_1^* = T, C_2^* = \sum_{l=1}^n r_l^* T$, compute $S_{\imath,\beta,2} = sH_1(id_{\imath,\beta}, 2)$ and $h = H_6(C_1^*, m, id_{\imath,\beta})$, randomly choose $\gamma^* \in \{0,1\}^{l_0+l_1+l_4}$, and compute $C_3^* = (id_{\imath,\beta}, m, hS_{\imath,\beta,2} + sT) \oplus \gamma^*$.

- Return $C^* = (C_1^*, C_2^*, C_3^*)$. Note that $\mathcal{A}$ cannot recognize that C is not a proper signcrypted message unless he queries H_5 on $\hat{e}(\sum_{l=1}^n (Q_{l,0}^* + f_l Q_{l,1}^*), sT)\hat{e}(v_\vartheta^* T, J^*)\hat{e}(P_1, \sum_{l=1}^n r_l T)\hat{e}(T, \alpha^{k+1}P)$.

Phase 2: The queries made by $\mathcal{A}$ in Phase 2 are responded in the same way as those made by $\mathcal{A}$ in **Phase 1**, except that $\mathcal{A}$ is not allowed to make a $\mathsf{Decrypt/Verify}(\Pi^{\pi'}_{\mathcal{U}'_i}, C^*)$ query, where $\Pi^{\pi'}_{\mathcal{U}'_i}$ and $\Pi^{\pi}_{\mathcal{U}_i}$ are partnered. In addition, the instance $\Pi^{\pi}_{\mathcal{U}_i}$ which $\mathcal{A}$ intend to attack must be kept fresh after these queries.

Response: Once $\mathcal{A}$ finishes querying, $\mathcal{A}$ returns his guess $\beta' \in \{0,1\}$. If $\beta' \neq \beta$, $\mathcal{B}$ aborts the game; otherwise $\mathcal{B}$ randomly chooses an item (μ_i, γ_i) from L_5 and returns the value

$$\mu_i(\hat{e}(\sum_{l=1}^n (Q_{l,0}^* + f_l Q_{l,1}^*), sT)\hat{e}(v_l^* T, J^*)\hat{e}(P_1, \sum_{l=1}^n r_l T))^{-1}$$

as its guess at the solution to the k-BDHE problem, where $Q_{l,0}^* = H_1(id_l^*, 0)$, $Q_{l,1}^* = H_1(id_l^*, 1)$, $J^* = \sum_{l=1}^n Z_l^* = \sum_{l=1}^n z_l^* P + P_k$.

It's easy to see that the overall advantage for $\mathcal{B}$ to solve the k-BDHE problem is at least $\frac{1-2\epsilon'}{q_{H_5}(q_{dk}+1)e}\epsilon$.

□

7. Conclusion

We have formalized the notion and security definitions of identity based AAGKAwSNP which allows a group of users to establish a secure group communication channel efficiently. Following this notion, we have proposed a concrete identity based AAGKAwSNP protocol. The protocol has been proven to achieve direct CCA secrecy, key escrow freeness, sender non-repudiation and sender privacy under the k-BDHE and CDH assumptions.

Acknowledgments

The following funding sources are gratefully acknowledged: NSF of China (grants 61572198, 61321064).

References

[1] D. Boneh, X. Boyen, and E. J. Goh, "Hierarchical identity based encryption with constant size ciphertext", *Proc. Annu. Int. Conf. Theory and Appl. of Cryptographic Techniques (TCC)*, pp. 440–456, 2005.

[2] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys", *Proc. 25th Annu. Int. Cryptology Conf. (CRYPTO'05)*, pp. 258–275, 2005.

[3] D. Boneh and M. Hamburg, "Generalized identity based and broadcast encryption schemes", *Proc. 14th Int. Conf. on the Theory and Appl. of Cryptology and Inform. Security (ASIACRYPT'08)*, pp. 455–470, 2008.

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

[5] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation", *Proc. 34th Annu. Int. Cryptology Conf. (CRYPTO'14)*, pp. 480–499, 2014.

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

[7] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys", *Proc. 13th Int. Conf. on the Theory and Appl. of Cryptology and Inform. Security (ASIACRYPT'07)*, pp. 200–215, 2007.

[8] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[9] R. Dutta and R. Barua, "Provably secure constant round contributory group key agreement in dynamic setting", *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2007–2025, 2008.

[10] F. Eiichiro and O. Tatsuaki, "Secure integration of asymmetric and symmetric encryption schemes", *J. Cryptology*, vol. 26, no. 1, pp. 80–101, 2013.

[11] A. Fiat and M. Naor, "Broadcast encryption", in *Proc. 13th Annu. Int. Cryptology Conf. (CRYPTO'93)*, pp. 480–491, 1994.

[12] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices", *Proc. 32nd Annu. Int. Conf. on the Theory and Appl. of Cryptographic Techn. (EUROCRYPT'13)*, pp. 1–17, 2013.

[13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits", *Proc. 54th IEEE Annu. Symp. on Foundations of Comput. Sci. (FOCS)*, pp. 40–49, 2013.

[14] C. Gentry and B. Waters, "Adaptive security in broadcast encryption systems (with short ciphertexts)", in *Proc. 28nd Annu. Int. Conf. on the Theory and Appl. of Cryptographic Techn. (EUROCRYPT'09)*, pp. 171–188, 2009.

[15] D. Halevy and A. Shamir, "The lsd broadcast encryption scheme", in *Proc. 22nd Annu. Int. Cryptology Conf. (CRYPTO'02)*, pp. 47–60, 2002.

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

[17] S. Jiang, "Group key agreement with local connectivity", *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 3, pp.326–339, 2016.

[18] A. Joux, "A one round protocol for tripartite diffie–hellman", *J. Cryptology*, vol. 17, no. 4, pp. 263–276, 2004.

[19] J. Kim, W. Susilo, M. Au, and J. Seberry, "Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext", *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 679–693, 2015.

[20] J. Li, L. Zhang, J. Liu, H. Qian, and Z. Dong "Privacy-Preserving Public Auditing Protocol for Low Performance End Devices in Cloud", *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2572–2583, 2016.

[21] B. Libert, K. G. Paterson, and E. A. Quaglia, "Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model", *Proc. 15th Int. Conf. on Practice and Theory in Public Key Cryptography (PKC'12)*, pp. 206–224, 2012.

[22] W. Liu, J. Liu, Q. Wu, B. Qin, and Y. Li, "Practical chosen-ciphertext secure hierarchical identity-based broadcast encryption", *Int. J. Inform. Security*, vol. 15, no. 1, pp. 35–50, 2016.

[23] 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.

[24] M. Naor and B. Pinkas, "Efficient trace and revoke schemes", *Proc. 4th Int. Conf. Financial Cryptography (FC)*, pp. 1–20, 2001.

[25] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures", *J. cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[26] A. Shamir, "Identity-based cryptosystems and signature schemes", *Advances in cryptology–CRYPTO '85*, pp. 47–53, 1985.

[27] W.-G. Tzeng and Z.-J. Tzeng, "Round-efficient conference key agreement protocols with provable security", *Proc. 6th Int. Conf. on the Theory and Appl. of Cryptology and Inform. Security (ASIACRYPT'00)*, pp. 614–627, 2000.

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

[29] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs", *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, 2000.

[30] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement", *Proc. 28th Annu. Int. Conf. on the Theory and Appl. of Cryptographic Techn. (EUROCRYPT'09)*, pp. 153–170, 2009.

[31] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, and J. A. Manjón, "Fast transmission to remote cooperative groups: A new key management paradigm", *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 621–633, 2013.

[32] C. Xu, Z. Li, Y. Mu, H. Guo, and T. Guo, "Affiliation-hiding authenticated asymmetric group key agreement", *The Comput. J.*, vol. 55, no. 10, pp. 1180–1191, 2012.

[33] C. Xu, L. Zhu, Z. Li, and F. Wang, "One-round affiliation-hiding authenticated asymmetric group key agreement with semi-trusted group authority", *The Comput. J.*, vol.58, no. 10, pp. 2509-2519, 2015.

[34] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya, "Id-based encryption for complex hierarchies with applications to forward security and

broadcast encryption", *Proc. 11th ACM conf. on Comput. and commun. security (CCS'04)*, pp. 354–363, 2004.

[35] X. Yi, F.Y. Rao, Z. Tari, F. Hao, E. Bertino, I. Khalil, and A.Y. Zomaya, "ID2S Password-Authenticated Key Exchange Protocols", *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3687–3701, 2016.

[36] W. Yu, Y. Sun, and K. R. Liu, "Optimizing rekeying cost for contributory group key agreement schemes", *IEEE Trans. Depend. Secure Comput.*, vol. 4, no. 3, pp. 228–242, 2007.

[37] Y. Yu, M. H. Au , G. Ateniese, X. Huang, W. Susilo, Y. Dai and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage", *IEEE Trans. Inf. Forensics Security*, vol. 12, no 4, pp. 767-778, 2017.

[38] 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 Int. Conf. on Inform. Security and Cryptology (INSCRYPT 2013)*, pp. 239–254, 2014.

[39] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, and Z. Dong, "Round-efficient and sender-unrestricted dynamic group key agreement protocol for secure group communications", *IEEE Trans. Inf. Forensics Security*, vol. 12, no.10, pp. 2352–2364, 2015.

[40] L. Zhang, Q. Wu, B. Qin, H. Deng, J. Li, J. Liu, and W. Shi, "Certificateless and identity-based authenticated asymmetric group key agreement", *Int. J. Information Security*, doi. 10.1007/s10207-016-0339-8.

[41] L. Zhang, Q. Wu, B. Qin, H. Deng, J. Liu, and W. Shi, "Provably secure certificateless authenticated asymmetric group key agreement", *Proc. 10th Int. Conf. Inform. Security Practice and Experience (ISPEC 2014)*, pp. 496–510, 2014.

[42] L. Zhang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based authenticated asymmetric group key agreement protocol", *Proc. 16th Annu. Int. Conf. Comput. and Combinatorics (COCOON 2010)*, pp. 510–519, 2010.

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

[44] L. Zhang, C. Hu, Q. Wu, J. Domingo-Ferrer, B. Qin, “Privacy-preserving vehicular communication authentication with hierarchical aggregation and fast response”, *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2562-2574, 2016.

[45] X. Zhao, D. Wei, and H. Wang, “Asymmetric group key agreement with traitor traceability”, *Proc. Int. Conf. on Computational Intelligence and Security (CIS)*, pp. 347–351, 2010.

[46] Z. Zhou, D. Huang, and Z. Wang, “Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption”, *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 126–138, 2015.