Control and Cybernetics

vol. **42** (2012) No. 2

An efficient provably secure certificateless aggregate signature applicable to mobile computation[*][†]

by

Hu Xiong[1], Qianhong Wu[2,3], and Zhong Chen[1]

[1]Key Laboratory of Network and Software Security Assurance, Institute of Software
School of Electronics Engineering and Computer Science, Peking University
Beijing, 100871, China
[2]School of Computer, Wuhan University, Wuhan, 430072, China
[3]Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics
Tarragona, E-43007, Catalonia
[4]State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China
e-mail: xionghu.uestc@gmail.com, qianhong.wu@urv.cat, chen@ss.pku.edu.cn

Abstract: An aggregate signature scheme allows a public algorithm to aggregate n signatures on n distinct messages from n signers into a single signature. By validating the single resulting signature, one can be convinced that the messages have been endorsed by all the signers. Certificateless aggregate signatures allow the signers to authenticate messages without suffering from the complex certificate management in the traditional public key cryptography or the key escrow problem in identity-based cryptography. In this paper, we present a new efficient certificateless aggregate signature scheme. Compared with up-to-date certificateless aggregate signatures, our scheme is equipped with a number of attracting features: (1) it is shown to be secure under the standard computational Diffie-Hellman assumption in the random oracle model; (2) the security is proven in the strongest security model so far; (3) the signers do not need to be synchronized; and (4) its performance is comparable to the most efficient up-to-date schemes. These features are desirable in a mobile networking and computing environment where the storage/computation capacity of the end devices are limited, and due to the wireless connection and distributed feature, the computing devices are easy to be attacked and hard to be synchronized.

Keywords: message authentication, certificateless cryptography, aggregate signature, mobile computing.

*Submitted: November 2011; Accepted: April 2012

†This paper is a full version of the paper due to Xiong et al. (2011) in the proceedings of INCoS 2011.

1. Introduction

As information and communications technologies (ICT) become increasingly pervasive, it is expectable for mobile devices to communicate with each other anywhere anytime, e.g. vehicular networks in which each vehicle may exchanges 500-2000 messages/second in a high-density traffic scenario (see Wu et al., 2010; or Zhang et al., 2010b). A basic requirement in these scenarios is that the exchanged messages have to be authentic "for the prevention, investigation, detection, and prosecution of serious criminal offences (see European Parliament, 2005)". Employing digital signatures is one of the main approaches to meet this requirement, involving the challenges (1) to guarantee that the authorship of the signatures is sound, (2) the signatures can be efficiently verified, and (3) the signatures can be efficiently stored for possible juristical witnesses. These challenges have to be addressed to extensively deploy signatures to secure such applications.

Related work. Public key infrastructure is a conventional approach to address the first challenge. In the traditional public key cryptography (PKC), the public key of a user is usually a "random" string that is unrelated to the identity of the user, so a trusted-by-all certificate authority (CA) is employed to assure the relationship between the cryptographic keys and the user. As a result, traditional PKC requires a large amount of storage space and computing time to manage the certificates.

To circumvent complicated certificate management, Shamir (1984) introduced the concept of ID-based public key cryptography (ID-PKC). In ID-PKC, the public key of each user is easily computable from a string corresponding to this user's identity, e.g., a telephone number, while the private key associated with that identity is computed and issued secretly to the user by a trusted private key generator (PKG). This property dramatically simplifies the certificate management in the traditional PKC. However, an inherent problem of ID-PKC is key escrow, i.e., the PKG knows users' private key. Clearly, a malicious PKG can decrypt any ciphertext and forge the signature of any user. Due to this inherent problem, ID-PKC is considered to be suitable only for close networks (see Shamir, 1984).

To avoid the shortcomings of traditional PKC and ID-PKC, Al-Riyami and Paterson (2003) proposed a paradigm called certificateless public key cryptography (CL-PKC) in 2003. The concept was introduced to suppress the inherent key-escrow property of ID-PKC without losing their most attractive advantage, i.e., relief from numerous digital certificates and their heavy management overhead. The basic idea of CL-PKC is that only the user has the authority to compute his full private key by combining his partial private key generated by a third party called Key Generation Center (KGC) and a secret value chosen by himself. Hence, the KGC does not have access to the user's full private key. The public key of a user is computed from the KGC's public parameters and the secret value of the user, and it is published by the user himself. Thus, CL-PKC avoids the complicated certificate management problem in traditional PKC, including revocation, storage and distribution and the computational cost of certificate verification which are particularly acute in processor or bandwidth-limited environments. It also avoids the key escrow problem which, restricts ID-PKC to small, closed groups or to applications with limited security requirements.

Although CL-PKC seems to be the most efficient approach to guarantee the authorship of signatures, it is not trivial to design secure certificateless signature (CLS) schemes. In the seminal CL-PKC paper of Al-Riyami and Paterson (2003), they presented a CLS scheme. Later, Huang et al. (2005) pointed out a security drawback of the original scheme and proposed a secure one. A generic construction of CLS scheme was proposed by Yum and Lee (2004) in ACISP 2004. However, Hu et al. (2006) showed that the Yum-Lee construction is insecure and proposed a fix in the standard model. In ACNS 2006, Zhang and Wong (2006) presented an efficient CLS scheme from pairings. Gorantla and Saxena (2005) introduced a new construction of CLS scheme without providing formal proofs. After that, several efficient and secure CLS schemes have been proposed (see Choi et al.,2007, 2011; Zhang and Zhang, 2008).

Regarding the formal security model, two types of adversaries, Type I and Type II adversaries, should be considered for the CL-PKC as defined by Al-Riyami and Paterson (2003). Type I adversary represents the normal third party attacker who has no access to the master key, but is allowed to replace the public keys of the users. Type II adversary represents the malicious KGC who is equipped with the master key, but is not able to replace public keys. According to their attack power, Type I/II adversaries can be further classified into three kinds of normal, strong, and super Type I/II adversaries, Huang et al. (2007). The normal Type I/II adversary can only obtain the valid signatures for the original public key. The strong Type I/II adversary can obtain the valid signatures for the public key replaced by himself if he additionally submits the secret value corresponding to the replaced public key. The super Type I/II adversary can obtain the valid signatures for the replaced public key, without additional submission. The last is the strongest security model of CLS schemes and it is also employed in this paper.

Recently, efforts have been devoted to fast signature verification and efficient signature storage. This is fulfilled by the advanced notion of aggregate signatures introduced by Boneh et al. (2003). In this notion, given n signatures (as well as the associating public keys) on n messages from n users, anyone can combine all of these signatures into a single one. The resulting signature can convince a verifier that the n users indeed signed the n corresponding messages. This feature greatly reduces the computational overhead to verify signatures and the communication/storage cost to relay/store signatures.

Since the Boneh et al. (2003) scheme, a number of aggregate signature schemes have been proposed in PKC (see Ahn et al., 2010; Bagherzandi and Jarecki, 2010; Boldyreva et al., 2007; Lu et al., 2006; Lysyanskaya et al., 2004;

Neven, 2008) and ID-PKC (see Cheng et al., 2005; Gentry and Ramzan, 2006; Herranz, 2006; Shim, 2010; Xu et al., 2005), respectively. To realize the merits of aggregate signature and CL-PKC simultaneously, the concept of certificateless aggregate signature (CL-AS) has been introduced and several CL-AS schemes have been proposed (see Castro and Dahab, 2007; Gong et al., 2007; Zhang and Zhang, 2009; Zhang et al., 2010a). Most CL-AS schemes are only provably secure against the strong Type I/II adversary (see Castro and Dahab, 2007; Gong et al., 2007; Zhang et al., 2010a). Only one scheme in Zhang and Zhang(2009) is provably secure against the super Type I/II adversary. Furthermore, the schemes (see Zhang and Zhang, 2009; Zhang et al., 2010a) require certain synchronization, i.e., all signers must share the same string based on synchronized clocks to generate the aggregate signature. One may observe that it is not easy to achieve synchronization in many mobile computing scenarios. Also, many schemes (see Castro and Dahab, 2007; Gong et al., 2007; Zhang and Zhang, 2009) require a large number of pairing operations and a long signature size, meaning a departure from the main goals of aggregate signatures.

Our contribution. In this paper, we propose an efficient CL-AS scheme and show its security in the random oracle model against the super Type I/II adversary. The proof is built on the standard Computational Diffie-Hellman assumption. The proposed scheme does not require synchronization among signers for aggregating randomness, which makes it more suitable for *ad hoc* networks. As for performance, our scheme allows multiple signers to sign multiple messages in an efficient way and the result of aggregate signature consists of only two groups elements, and the verification procedure needs only a very small constant number of pairing operations. The performance is comparable to the most efficient schemes so far. These features show, that our scheme is secure and practical for mobile applications where the end devices have only limited computation and storage capacities.

Paper organization. The rest of this paper is organized as follows. A brief review of some basic concepts and security notions used in our scheme is given in Section 2. In Section 3, we construct an efficient CL-AS scheme and provide its security proof. Finally, the conclusions are given in Section 4.

2. Preliminaries

In this section, we will review some background required in this paper, namely bilinear pairing and the definition of a CL-AS scheme. The notations used throughout the paper are listed in Table 1.

2.1. Bilinear pairing and complexity assumption

Following the notions of Wu et al. (2009, 2011), we briefly review bilinear parings. Let $\mathbb{G}_1$ denote an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same order. Let P be a generator of $\mathbb{G}_1$, and $\hat{e}$ be a

Table 1. Notations

Notations	Descriptions
CL-AS:	**C**ertificate**l**ess **A**ggregate **S**ignature
KGC:	**K**ey **G**eneration **C**enter
$\mathbb{G}_1$:	An additive group of prime order q
$\mathbb{G}_2$:	A multiplicative group of prime order q
$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$:	An efficiently computable bilinear map
P, Q, Q'	Three different generators in $\mathbb{G}_1$
k:	A security parameter
$\mathcal{A}_1, \mathcal{A}_2$:	A type I/II adversary
ID_i	The identity of a user
psk_{ID_i}	The partial private key of a user with identity ID_i
usk_{ID_i}, upk_{ID_i}	The user secret key and user public key of a user with identity ID_i, respectively
m_i :	A message to be signed
H_i :	A hash function
σ_i :	A single signature on a message
σ :	An aggregate signature

bilinear map such that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, with the following properties:

1. Bilinearity: for all $P, Q \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_q$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$.
3. Computability: it is efficient to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

The security of our signature scheme will be reduced to the hardness of the Computational Diffe-Hellman (CDH) problem in the group, in which the signature is constructed. We briefly review the definition of the CDH problem:

Definition 1 *Given the elements P, aP and bP for some random values $a, b \in \mathbb{Z}_q$ the Computational Diffe-Hellman (CDH) problem consists of computing the element abP.*

The success probability of an algorithm $\mathcal{A}$ in solving CDH problem in $\mathbb{G}_1$ is defined as

$$\mathrm{Succ}^{CDH}_{\mathcal{A},\mathbb{G}_1} = Pr[\mathcal{A}(P, aP, bP) = abP : a, b \in \mathbb{Z}_q]. \quad (1)$$

The CDH assumption states that for every probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathrm{Succ}^{CDH}_{\mathcal{A},\mathbb{G}_1}$ is negligible.

2.2. Security notions

Components of Certificateless Aggregate Signature Scheme. A certificateless Aggregate Signature (CL-AS) scheme CL-AS= (MasterKeyGen, PartialKeyGen, UserKeyGen, Sign, Aggregate and Aggregate Verify) is specified by six polynomial time algorithms with the following functionality:

1. The randomized parameter generation algorithm $\mathsf{MasterKeyGen}$ takes as input 1^k, where k is the security parameter, and outputs a master public/secret key pair (mpk, msk). The algorithm is assumed to be run by a Key Generation Center (KGC) for the initial setup of a CL-AS scheme.
2. The randomized private key generation algorithm $\mathsf{PartialKeyGen}$ takes as input msk and user's identity $ID_i \in \{0,1\}^*$ and generates a key psk_{ID_i} called partial private key. This algorithm is run by the KGC once for each user, and the partial private key is assumed to be distributed securely to the corresponding user.
3. The randomized user key generation algorithm $\mathsf{UserKeyGen}$ takes as input mpk and user's identity ID_i and generates a user public/secret key pair (upk_{ID_i}, usk_{ID_i}). This algorithm is supposed to be run by each user in the system.
4. The randomized signing algorithm Sign is run by a signer ID_i that takes as input a signing key (psk_{ID_i}, usk_{ID_i}), a message $m_i \in \{0,1\}^*$ and outputs a signature $\sigma_i \leftarrow \mathsf{Sign}(psk_{ID_i}, usk_{ID_i}, m_i)$.
5. The randomized aggregating algorithm $\mathsf{Aggregate}$ takes as input an aggregating set U of n users $\{\mathfrak{U}_1, \cdots, \mathfrak{U}_n\}$ with the identity ID_i and the corresponding public key upk_{ID_i} of each user $\mathfrak{U}_i$, and a signature σ_i on a message m_i under identity ID_i and the public key upk_{ID_i} for each user $\mathfrak{U}_i \in U$. The output of this algorithm is an aggregate signature σ on messages $\{m_1, \cdots, m_n\}$.
6. The randomized verification algorithm $\mathsf{Aggregate\ Verify}$ takes as input mpk, an aggregating set U of n users $\{\mathfrak{U}_1, \cdots, \mathfrak{U}_n\}$ with the identity ID_i and the corresponding public key upk_{ID_i} of each user $\mathfrak{U}_i$, an aggregate signature σ on messages $\{m_1, \cdots, m_n\}$. It outputs True if the signature is correct, or $\perp$ otherwise.

The Adversaries Model of the CL-AS Scheme. Combining the security notions of CL-PKC and security models of aggregate signature schemes in traditional PKC and ID-PKC, two types of security, Type-I security and Type-II security, for CL-AS scheme along with two types of adversaries, $\mathcal{A}_1$ and $\mathcal{A}_2$, have been defined, Choi et al. (2011), Zhang and Zhang (2009), Zhang et al. (2010a). Adversary $\mathcal{A}_1$ models a malicious adversary which compromises the user secret key usk_{ID} or replaces the user public key upk_{ID}, but cannot compromise the master secret key msk nor get access to the partial private key psk_{ID}. Adversary $\mathcal{A}_2$ models the malicious-but-passive KGC who controls the generation of the master public/secret key pair, and that of any partial private key psk_{ID}. The following are the five oracles which can be accessed by the adversaries.

1. **CreateUser**: On input of an identity $ID_i \in \{0,1\}^*$, if ID_i has already been created, nothing is to be carried out. Otherwise, the oracle generates $psk_{ID_i} \leftarrow \text{PartialKeyGen}(msk, ID_i)$ and $(upk_{ID_i}, usk_{ID_i}) \leftarrow$ UserKeyGen (mpk, ID_i). It then stores $(ID_i,\ psk_{ID_i},\ upk_{ID_i},\ usk_{ID_i})$ in a list L. In

both cases, upk_{ID_i} is returned.

2. **RevealPartialKey**: On input of an identity ID_i, the oracle searches L for an entry corresponding to ID_i. If it is not found, $\perp$ is returned; otherwise, the corresponding psk_{ID_i} is returned.
3. **RevealSecretKey**: On input of an identity ID_i, the oracle searches L for an entry corresponding to ID_i. If it is not found, $\perp$ is returned; otherwise, the corresponding usk_{ID_i} is returned.
4. **ReplaceKey**: On input out an identity ID_i and a user public/secret key pair $(upk^*_{ID_i}, usk^*_{ID_i})$, the oracle searches L for the entry of ID_i. If it is not found, nothing will be carried out. Otherwise, the oracle updates $(ID, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ to $(ID, psk_{ID_i}, upk^*_{ID_i}, usk^*_{ID_i})$.
5. **Super-Sign**: On input of a message $m_i \in \{0,1\}^*$ for ID_i, the signing oracle returns a valid signature σ_i such that $\mathsf{True} \leftarrow \mathsf{Verify}(mpk, ID_i, upk_{ID_i}, m_i, \sigma_i)$, where upk_{ID_i} is the current public key corresponding to ID_i and it may be replaced by the **ReplaceKey** query.

Note that when the oracle **ReplaceKey** is queried, $usk^*_{ID_i}$ can be an empty string. In this case, it means that the user secret key is not provided. If $usk^*_{ID_i}$ is an empty string and the original user secret key of an identity ID_i is replaced with $usk^*_{ID_i}$, then the empty string will be returned if the **RevealSecretKey** oracle is queried on ID_i. Also note that even if $usk^*_{ID_i}$ is not an empty string, it does not mean that $usk^*_{ID_i}$ is the corresponding secret key of $upk^*_{ID_i}$. Hence, as mentioned, the signature generated by the signing oracle **Super-Sign** will be a valid signature under the replaced user public key $upk^*_{ID_i}$.

We define two games, one for $\mathcal{A}_1$ and the other one for $\mathcal{A}_2$.

Game I: Let $\mathcal{S}_1$ be the simulator/challenger game and $k \in \mathbb{N}$ be a security parameter.

1. $\mathcal{S}_1$ executes $\mathsf{MasterKeyGen}(1^k)$ to get (mpk, msk).
2. $\mathcal{S}_1$ runs $\mathcal{A}_1$ on 1^k and mpk. During the simulation, $\mathcal{A}_1$ can make queries onto the oracles **CreateUser**, **RevealPartialKey**, **RevealSecretKey**, **ReplaceKey** and **Super-Sign**.
3. $\mathcal{A}_1$ outputs a set of n users with identities from the set $L^*_{ID} = \{ID^*_1, \cdots, ID^*_n\}$ and corresponding public keys from the set $L^*_{upk} = \{upk^*_1, \cdots, upk^*_n\}$, n messages $L^*_m = \{m^*_1, \cdots, m^*_n\}$ and an aggregate signature σ^*.

We say that $\mathcal{A}_1$ wins **Game I**, iff

- σ^* is a valid aggregate signature on messages $\{m^*_1, \cdots, m^*_n\}$ under identities $\{ID^*_1, \cdots, ID^*_n\}$ and the corresponding public keys $\{upk^*_1, \cdots, upk^*_n\}$.
- At least one of the identities, without loss of generality, say $ID^*_1 \in L^*_{ID}$, has not been submitted during the **RevealPartialKey**(ID^*_1) queries to get the user partial key $psk^*_{ID_1}$. Also, the oracle **Super-Sign** has never been queried with (ID^*_1, m^*_1).

Definition 2 *A CL-AS scheme is said to be Type-I secure if there is no probabilistic polynomial-time adversary $\mathcal{A}_1$ winning **Game I** with non-negligible advantage.*

Game II: Let $\mathcal{S}_2$ be the game challenger and $k \in \mathbb{N}$ be a security parameter. There are two phases of interactions between $\mathcal{S}_2$ and $\mathcal{A}_2$.

1. $\mathcal{S}_2$ executes $\mathcal{A}_2$ on input 1^k, which returns a master public/secret key pair (mpk, msk) to $\mathcal{A}_2$. Note that $\mathcal{A}_2$ cannot make any query at this stage.
2. During this stage of simulation, $\mathcal{A}_2$ can make queries onto oracle **RevealSecretKey** and **Super-Sign**. $\mathcal{A}_2$ can also make queries to **CreateUser**. Note that oracle **RevealPartialKey** is not accessible and no longer needed as $\mathcal{A}_2$ has the master secret key, and when $\mathcal{A}_2$ issues a query to **CreateUser** oracle, it has to additionally provide the partial private key psk_{ID}.
3. At the end of this phase, $\mathcal{A}_2$ is to output a set of n users whith identities from the set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$ and corresponding public keys from the set $L_{upk}^* = \{upk_1^*, \cdots, upk_n^*\}$, n messages $L_m^* = \{m_1^*, \cdots, m_n^*\}$ and an aggregate signature σ^*.

We say that $\mathcal{A}_2$ wins **Game II**, iff

- σ^* is a valid aggregate signature on messages $\{m_1^*, \cdots, m_n^*\}$ under identities $\{ID_1^*, \cdots, ID_n^*\}$ and the corresponding public keys $\{upk_1^*, \cdots, upk_n^*\}$.
- At least one of the identities, without loss of generality, say $ID_1^* \in L_{ID}^*$, has not been submitted during the **RevealSecretKey**(ID_1^*) queries to get the user secret key $usk_{ID_1}^*$. Also, the oracle **Super-Sign** has never been queried with (ID_1^*, m_1^*).

Definition 3 *A CL-AS scheme is said to be Type-II secure if there is no probabilistic polynomial-time adversary $\mathcal{A}_2$ winning **Game II** with non-negligible advantage.*

3. New certificateless aggregate signature

3.1. The proposed certificateless aggregate signature scheme

In this section, we will give the concrete construction of our CL-AS scheme. Compared to the preliminary version (see Xiong et al., 2011), randomness has been added to the signature generation process to fix a flaw and guarantee strong security. The proposed scheme comprises the following algorithms.

MasterKeyGen: Given a security parameter $k \in \mathbb{Z}$, the algorithm works as follows:

1. Run the parameter generator on input k to generate a prime q, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order q, three different generators P, Q and Q' in $\mathbb{G}_1$ and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
2. Select a master-key $s \in_R \mathbb{Z}_q^*$ and sets $P_{pub} = sP$.
3. Choose cryptographic hash functions $H_0, H_0' : \{0,1\}^* \to \mathbb{G}_1$ and $H_1, H_2, H_2' : \{0,1\}^* \to \mathbb{Z}_q^*$. The security analysis will review H_1 and H_2 as random oracles. The master-key is s. The system parameters are $\{q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, Q', P_{pub}, H_0, H_0', H_1, H_2, H_2'\}$.

PartialKeyGen: Given a user's identity $ID_i \in \{0,1\}^*$, KGC first computes $Q_{ID_i} = H_0(ID_i)$ and $Q'_{ID_i} = H'_0(ID_i)$. It then sets this user's partial key $psk_{ID_i} = (sQ_{ID_i}, sQ'_{ID_i})$ and transmits it to user ID_i secretly.

UserKeyGen: The user ID_i selects a secret value $x_{ID_i} \in_R \mathbb{Z}_q^*$ as his secret key usk_{ID_i}, and computes his public key as $upk_{ID_i} = x_{ID_i}P$.

Sign: Given its signing key (usk_{ID_i}, psk_{ID_i}), and a message $m_i \in \{0,1\}^*$, the signer, whose identity is ID_i, and the corresponding public key is upk_{ID_i} performs the following steps:

1. Compute $h_{i1} = H_1(m_i, ID_i, upk_{ID_i})$, $h_{i2} = H_2(m_i, ID_i, upk_{ID_i})$ and $h'_{i2} = H'_2(m_i, ID_i, upk_{ID_i})$.
2. Choose $r_i \in_R \mathbb{Z}_q^*$ and compute $R_i = r_iP$.
3. Compute $\sigma_i = h_{i1} \cdot x_{ID_i} \cdot Q + r_i \cdot Q' + h_{i2} \cdot sQ_{ID_i} + h'_{i2} \cdot sQ'_{ID_i}$.
4. Output (R_i, σ_i) as the signature on m_i.

Aggregate: Anyone can act as an aggregate signature generator and aggregate a collection of individual signatures. For an aggregating set of n users $\{\mathfrak{U}_1, \cdots, \mathfrak{U}_n\}$ with identities $\{ID_1, \cdots, ID_n\}$, the corresponding public keys $\{upk_1, \cdots, upk_n\}$, and message-signature pairs $(m_1, R_1, \sigma_1), \cdots, (m_n, R_n, \sigma_n)$ from $\{\mathfrak{U}_1, \cdots, \mathfrak{U}_n\}$, respectively, the aggregate signature generator computes $R = \sum_{i=1}^n R_i$, $\sigma = \sum_{i=1}^n \sigma_i$ and outputs (R, σ) as the aggregate signature.

Aggregate Verify: To verify an aggregate signature (R, σ) signed by n users $\{\mathfrak{U}_1, \cdots, \mathfrak{U}_n\}$ with identities $\{ID_1, \cdots, ID_n\}$ and the corresponding public keys $\{upk_1, \cdots, upk_n\}$ on messages $\{m_1, \cdots, m_n\}$, the verifier performs the following steps:

1) Compute $Q_{ID_i} = H_0(ID_i)$, $Q'_{ID_i} = H'_0(ID_i)$, $h_{i1} = H_1(m_i, ID_i, upk_{ID_i})$, $h_{i2} = H_2(m_i, ID_i, upk_{ID_i})$ and $h'_{i2} = H'_2(m_i, ID_i, upk_{ID_i})$ for $i = 1, \cdots, n$.

2) Verify the equation

$$\begin{aligned} \hat{e}(\sigma, P) &= \hat{e}(\sum_{i=1}^n h_{i1} upk_{ID_i}, Q)\hat{e}(R, Q') \\ &\quad \times \hat{e}(\sum_{i=1}^n (h_{i2}Q_{ID_i} + h'_{i2}Q'_{ID_i}), P_{pub}). \end{aligned} \tag{2}$$

If it holds, accept the signature; else reject it.

3.2. Security proof

Assuming that the CDH problem is hard, we now show the security of our CL-AS scheme.

THEOREM 1 *In the random oracle model, our CL-AS scheme is existentially unforgeable against adaptive chosen-message attacks under the assumption that the CDH problem in $\mathbb{G}_1$ is intractable.*

The theorem follows at once from Lemmas 1 and 2, according to Definitions 2 and 3.

LEMMA 1 *If a probabilistic polynomial-time forger $\mathcal{A}_1$ has an advantage ε in forging a signature in an attack modeled by **Game I** of Definition 2 after running in time t and making q_{H_0} queries to random oracles H_0 and H_0', q_{H_1} queries to random oracle H_1, q_{H_2} queries to random oracles H_2 and H_2', q_{CreU} queries to the **CreateUser** request oracle, q_{RPar} queries to the **RevealPartialKey** extraction oracle, q_{RSec} queries to the **RevealSecretKey** extraction oracle, and q_{SupSig} queries to the **Super-Sign** oracle, then the CDH problem can be solved with probability $\varepsilon' > (1 - \frac{1}{q_{H_0}})^{q_{RPar}+n-1} q_{H_0} \varepsilon$.*

Proof. Let $(X = aP, Y = bP)$ be a random instance of the CDH problem in $\mathbb{G}_1$. Here, P is a generator of $\mathbb{G}_1$ with prime order q, and the elements a, b are taken uniformly at random from $\mathbb{Z}_q^*$. By using the forgery algorithm $\mathcal{A}_1$, we will construct an algorithm $\mathcal{S}_1$ which outputs the CDH solution abP in $\mathbb{G}_1$. ■

Algorithm $\mathcal{S}_1$ chooses two random $t, t' \in \mathbb{Z}_q^*$, and sets $P_{pub} = X$, $Q = tP$ and $Q' = t'P$, and then starts performing oracle simulation. Without loss of generality, we assume that, for any key extraction or signature query involving an identity, an $H_0(\cdot)$ and $H_0'(\cdot)$ oracle query has previously been made on that identity. And $\mathcal{S}_1$ maintains a list $L = \{(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})\}$, while $\mathcal{A}_1$ is making queries throughout the game. $\mathcal{S}_1$ responds to $\mathcal{A}_1$'s oracle as follows.

Queries on Oracle H_0, H_0': Suppose $\mathcal{A}_1$ makes at most q_{H_0} queries to H_0, H_0' oracle. First, $\mathcal{S}_1$ chooses $j \in [1, q_{H_0}]$ randomly. When $\mathcal{A}_1$ makes an H_0, H_0' query on ID_i, where $1 \le i \le q_{H_0}$, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{S}_1$ picks two random $\alpha, \beta \in \mathbb{Z}_q^*$ and returns $(Q_{ID_i} = Y, Q'_{ID_i} = \alpha(\beta P - Y))$. Then $\mathcal{S}_1$ inserts a tuple $(ID_i, Q_{ID_i}, Q'_{ID_i}, \alpha, \beta)$ in a list L_0. Otherwise, $\mathcal{S}_1$ picks two random $r_i, r_i' \in \mathbb{Z}_q^*$ and returns $(Q_{ID_i} = r_i P, Q'_{ID_i} = r_i' P)$, and adds $(ID_i, Q_{ID_i}, Q'_{ID_i}, r_i, r_i')$ to L_0.

Queries on Oracle H_1: Suppose (m_i, ID_i, upk_{ID_i}) is submitted to oracle $H_1(\cdot)$. $\mathcal{S}_1$ first scans $L_1 = \{(m_i, ID_i, upk_{ID_i}, h_{i1})\}$ to check whether H_1 has already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{S}_1$ picks at random $h_{i1} \in \mathbb{Z}_q^*$ and returns h_{i1} as a hash value of $H_1(m_i, ID_i, upk_{ID_i})$ to $\mathcal{A}_1$ and also stores the values in the list L_1.

Queries on Oracle H_2, H_2': Suppose (m_i, ID_i, upk_{ID_i}) is submitted to oracle $H_2(\cdot)$ and $H_2'(\cdot)$. $\mathcal{S}_1$ first scans $L_2 = \{(m_i, ID_i, upk_{ID_i}, h_{i2}, h_{i2}')\}$ to check whether H_2 and H_2' have already been defined for that input. If so, the previously defined value is returned. Otherwise, if $ID_i = ID^*$, $\mathcal{S}_1$ randomly picks at random $h_{i2} \in \mathbb{Z}_q^*$ and computes $h_{i2}' = h_{i2}\alpha^{-1}$; else if $ID_i \ne ID^*$, $\mathcal{S}_1$ randomly picks at random $h_{i2}, h_{i2}' \in \mathbb{Z}_q^*$. $\mathcal{S}_1$ returns h_{i2} and h_{i2}' as the hash values of $H_2(m_i, ID_i, upk_{ID_i})$ and $H_2'(m_i, ID_i, upk_{ID_i})$ to $\mathcal{A}_1$ and also stores the values in the list L_2.

RevealPartialKey Oracle: Suppose the request is on an identity ID_i. If $ID_i \ne ID^*$, $\mathcal{S}_1$ recovers the corresponding $(ID_i, Q_{ID_i}, Q'_{ID_i}, r_i, r_i')$ from the list L_0 and returns $psk_{ID_i} = (r_i X, r_i' X)$. Otherwise $\mathcal{S}_1$ outputs "failure" and halts because it is unable to coherently answer the query.

CreateUser Oracle: Suppose the request is on an identity ID_i.

- If the list L contains $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ checks whether $upk_{ID_i} = \perp$. If $upk_{ID_i} \neq \perp$, $\mathcal{S}_1$ returns upk_{ID_i} to $\mathcal{A}_1$. Otherwise, $\mathcal{S}_1$ randomly chooses $\nu_i \in \mathbb{Z}_q^*$ and sets $upk_{ID_i} = \nu_i P$ and $usk_{ID_i} = \nu_i$. $\mathcal{S}_1$ returns upk_{ID_i} to $\mathcal{A}_1$ and saves (upk_{ID_i}, usk_{ID_i}) into the list L.
- If the list L does not contain $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ sets $psk_{ID_i} = \perp$, and then randomly chooses $\nu_i \in \mathbb{Z}_q^*$ and sets $upk_{ID_i} = \nu_i P$ and $usk_{ID_i} = \nu_i$. $\mathcal{S}_1$ returns upk_{ID_i} to $\mathcal{A}_1$ and adds $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ to list L.

RevealSecretKey Oracle: Suppose the request is on an identity ID_i.

- If the list L contains $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ checks whether $usk_{ID_i} = \perp$. If $usk_{ID_i} \neq \perp$, $\mathcal{S}_1$ returns usk_{ID_i} to $\mathcal{A}_1$. Otherwise, $\mathcal{S}_1$ makes a **CreateUser** query itself to generate $(upk_{ID_i} = \nu_i P, usk_{ID_i} = \nu_i)$. Then $\mathcal{S}_1$ saves these values in the list L and returns $usk_{ID_i} = \nu_i$ to $\mathcal{A}_1$.
- If the list L does not contain $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ makes a **CreateUser** query itself, and then adds $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ to the list L and returns usk_{ID_i}.

ReplaceKey Oracle: Suppose $\mathcal{A}_1$ makes the query with (ID_i, upk'_{ID_i}).

- If the list L contains an element $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ sets upk_{ID_i}
$= upk'_{ID_i}$ and $usk_{ID_i} = \perp$.
- If the list L does not contain an item $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$, $\mathcal{S}_1$ sets $psk_{ID_i} = \perp$, $upk_{ID_i} = upk'_{ID_i}$ and $usk_{ID_i} = \perp$, and adds an element $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ to L.

Super-Sign Oracle: When $\mathcal{A}_1$ makes a **Super-Sign** query on m_i with ID_i, $\mathcal{S}_1$ first finds the corresponding $(ID_i, Q_{ID_i}, Q'_{ID_i}, r_i, r'_i)$ and $(ID_i, upk_{ID_i}, usk_{ID_i})$ from the lists L_0 and L, respectively. Then, $\mathcal{S}_1$ performs as follows:

- If $ID_i = ID^*$, $\mathcal{S}_1$ first checks whether ID_i has been in L_1 and L_2. If it is the case, $\mathcal{S}_1$ computes $\sigma_i = h_{i1} \cdot t \cdot upk_{ID_i} + t' \cdot R_i + h_{i2}\beta X$ and returns (R_i, σ_i). Else $\mathcal{S}_1$ picks two random $h_{i1}, h_{i2} \in \mathbb{Z}_q^*$, $R_i \in \mathbb{G}_1$ and computes $h'_{i2} = h_{i2}\alpha^{-1}$. $\mathcal{S}_1$ adds $(m_i, ID_i, upk_{ID_i}, h_{i1})$ and $(m_i, ID_i, upk_{ID_i}, h_{i2}, h'_{i2})$ to L_1, L_2, respectively. Finally, $\mathcal{S}_1$ computes $\sigma_i = h_{i1} \cdot t \cdot upk_{ID_i} + t' \cdot R_i + h_{i2}\beta X$ and returns (R_i, σ_i).
- Otherwise, $\mathcal{S}_1$ first checks whether ID_i has been in L_1 and L_2. If they are in the list, $\mathcal{S}_1$ computes $\sigma_i = h_{i1} \cdot t \cdot upk_{ID_i} + t' \cdot R_i + h_{i2} r_i X + h'_{i2} r'_i X$. $\mathcal{S}_1$ then returns (R_i, σ_i). Else, if ID_i is not on the lists L_1 or L_2, $\mathcal{S}_1$ picks three random $h_{i1}, h_{i2}, h'_{i2} \in \mathbb{Z}_q^*$, $R_i \in \mathbb{G}_1$ and adds $(m_i, ID_i, upk_{ID_i}, h_{i1})$ and $(m_i, ID_i, upk_{ID_i}, h_{i2}, h'_{i2})$ to L_1, L_2, respectively. Finally $\mathcal{S}_1$ computes $\sigma_i = h_{i1} \cdot t \cdot upk_{ID_i} + t' \cdot R_i + h_{i2} r_i X + h'_{i2} r'_i X$ and returns (R_i, σ_i).

Eventually, $\mathcal{A}_1$ is to output a set of n users whith identities from the set $L^*_{ID} = \{ID^*_1, \cdots, ID^*_n\}$ and corresponding public keys from the set $L^*_{upk} = \{upk^*_1, \cdots, upk^*_n\}$, n messages $L^*_m = \{m^*_1, \cdots, m^*_n\}$ and an aggregate signature

σ^*. It is required that there exist $ID^* \in \{ID_1^*, \cdots, ID_n^*\}$ such that $\mathcal{A}_1$ has not asked the partial private key. Without loss of generality, we assume that $ID^* = ID_1^*$. And $\mathcal{A}_1$ has not made an (m_1^*, ID_1^*) query to **Super-Sign** oracle. Furthermore, the aggregate signature (R^*, σ^*) should satisfy the aggregate verification as follows:

$$\begin{aligned}\hat{e}(\sigma^*, P) &= \hat{e}(R^*, Q')\hat{e}(\sum_{i=1}^{n} h_{i1}^* upk_{ID_i^*}, Q) \\ &\quad \times \hat{e}(\sum_{i=1}^{n}(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), P_{pub})\end{aligned} \tag{3}$$

Then, $\mathcal{S}_1$ finds the corresponding tuples $(ID_i^*, Q_{ID_i^*}, Q'_{ID_i^*}, r_i, r'_i)$ for $2 \leq i \leq n$ and $(ID_1^*, Q_{ID_1^*}, Q'_{ID_1^*}, \alpha, \beta)$ from the list L_0 respectively. The public keys $upk_{ID_i^*}$ may be replaced by $\mathcal{A}_1$. The following equation holds because the aggregate signature is valid.

$$\begin{aligned}\hat{e}(\sigma^*, P) &= \hat{e}(R^*, Q')\hat{e}(\sum_{i=1}^{n} h_{i1}^* upk_{ID_i^*}, Q)\hat{e}(\sum_{i=1}^{n}(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), P_{pub}) \\ &= \hat{e}(\sum_{i=1}^{n} h_{i1}^* upk_{ID_i^*}, tP)\hat{e}(\sum_{i=2}^{n}(h_{i2}^* r_i P + h_{i2}'^* r'_i P), X)\hat{e}(R^*, t'P) \\ &\quad \times \hat{e}(h_{12}^* Y + h_{12}'^* \alpha(\beta P - Y), X) \\ &= \hat{e}(\sum_{i=1}^{n} t h_{i1}^* upk_{ID_i^*}, P)\hat{e}(\sum_{i=2}^{n}(h_{i2}^* r_i + h_{i2}'^* r'_i)X, P)\hat{e}((h_{12}^* - h_{12}'^* \alpha)Y, X) \\ &\quad \times \hat{e}(h_{12}'^* \alpha\beta X, P)\hat{e}(t'R^*, P).\end{aligned} \tag{4}$$

Finally, $\mathcal{S}_1$ outputs abP as a solution to the CDH instance by computing

$$\begin{aligned}abP &= (h_{12}^* - h_{12}'^* \alpha)^{-1}(\sigma^* - \sum_{i=1}^{n} t h_{i1}^* upk_{ID_i^*} - \sum_{i=2}^{n}(h_{i2}^* r_i + h_{i2}'^* r'_i)X \\ &\quad - h_{12}'^* \alpha\beta X - t'R^*)\end{aligned} \tag{5}$$

This completes the description of $\mathcal{S}_1$. It remains to show that $\mathcal{S}_1$ solves the given instance of the CDH problem with probability of at least ε'. To do so, we analyze three events needed for $\mathcal{S}_1$ to succeed:

- E_1 : $\mathcal{S}_1$ does not abort as a result of any of $\mathcal{A}_1$'s **RevealPartialKey** queries.
- E_2 : $\mathcal{A}_1$ generates a valid and non-trivial aggregate signature forgery.
- E_3 : Event E_2 occurs, $ID_1^* = ID^*$ and $ID_i^* \neq ID^*$ for all $i, 2 \leq i \leq n$.

$\mathcal{S}_1$ succeeds if all of these events happen. The probability $Pr[E_1 \wedge E_2 \wedge E_3]$ is decomposed as $Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid E_1 \wedge E_2]$

The following claims give a lower bound for each of these terms.

Claim. The probability that algorithm $\mathcal{S}_1$ does not abort as a result of any of $\mathcal{A}_1$'s **RevealPartialKey** queries is at least $(1 - \frac{1}{q_{H_0}})^{q_{RPar}}$.

Proof. As $Pr[E_1] = 1 - \frac{1}{q_{H_0}}$, for a partial private key extraction query, the probability that $\mathcal{S}_1$ does not abort is $1 - \frac{1}{q_{H_0}}$. Since $\mathcal{A}_1$ makes at most q_{RPar} queries to the partial private key extraction oracle, the probability that $\mathcal{S}_1$ does not abort as a result of $\mathcal{A}_1$'s partial private key extraction queries is at least $(1 - \frac{1}{q_{H_0}})^{q_{RPar}}$. ■

Claim. If $\mathcal{S}_1$ does not abort as a result of $\mathcal{A}_1$'s **Super-Sign** queries and **RevealPartialKey** queries, then $\mathcal{A}_1$'s view is identical to its view in the attack. Hence, $Pr[E_2 \mid E_1] \geq \varepsilon$.

Claim. The probability that algorithm $\mathcal{S}_1$ does not abort after $\mathcal{A}_1$ outputs a valid and nontrivial forgery is at least $q_{H_0}(1 - \frac{1}{q_{H_0}})^{n-1}$. Hence, $Pr[E_3 \mid E_1 \wedge E_2] \geqslant q_{H_0}(1 - \frac{1}{q_{H_0}})^{n-1}$.

Proof. Algorithm $\mathcal{S}_1$ succeeds only if $\mathcal{A}_1$ generates a forgery such that $ID_1^* = ID^*$ and $ID_i^* \neq ID^*$ for all $i, 2 \leq i \leq n$. Hence, $Pr[E_3 \mid E_1 \wedge E_2] \geqslant q_{H_0}(1 - \frac{1}{q_{H_0}})^{n-1}$. ■

Therefore, the $\mathcal{S}_1$'s advantage in solving the CDH problem in $\mathbb{G}_1$ is at least $(1 - \frac{1}{q_{H_0}})^{q_{RPar}+n-1} q_{H_0}\varepsilon$.

LEMMA 2 *If a probabilistic polynomial-time forger $\mathcal{A}_2$ has an advantage ε in forging a signature in an attack modeled by* ***Game II*** *of Definition 3 after running in time t and making q_{H_2} queries to random oracles H_2, q_{CreU} queries to the* ***CreateUser*** *request oracle, q_{RSec} queries to the* ***RevealSecretKey*** *extraction oracle, and q_{SupSig} queries to the* ***Super-Sign*** *oracle, then the CDH problem can be solved with probability $(1 - \frac{1}{q_{CreU}})^{q_{SupSig}+q_{RSec}+n-1} q_{CreU}\varepsilon$.*

Proof. Suppose $\mathcal{A}_2$ is a **Type II** adversary that (t, ε)-breaks our CL-AS scheme. We show how to construct a t'-time algorithm $\mathcal{S}_2$ that solves the CDH problem on $\mathbb{G}_1$ with probability of at least ε'. Let $(X = aP, Y = bP) \in \mathbb{G}_1 \times \mathbb{G}_1$ be a random instance of the CDH problem taken as input by $\mathcal{S}_2$. ■

$\mathcal{S}_2$ randomly chooses $s \in \mathbb{Z}_q^*$ as the master key, and then initializes $\mathcal{A}_2$ with $P_{pub} = sP$ and also the master key s. After that, $\mathcal{S}_2$ sets $Q = X$, and chooses random $t' \in \mathbb{Z}_q^*$, and sets $Q' = t'P$. The adversary $\mathcal{A}_2$ then starts making oracle queries as described in Definition 3. Note that the user's partial key $psk_{ID_i} = (sH_0(ID_i), sH_0'(ID_i))$ can be computed by both $\mathcal{S}_2$ and $\mathcal{A}_2$, thus the hash functions $H_0(\cdot)$ and $H_0'(\cdot)$ are not modelled as a random oracle in this case.

CreateUser Oracle: Suppose $\mathcal{A}_2$ makes at most q_{CreU} queries to **CreateUser oracle**. First, $\mathcal{S}_2$ chooses $j \in [1, q_{CreU}]$ randomly. When $\mathcal{A}_2$ makes a **CreateUser** query on ID_i, where $1 \le i \le q_{CreU}$, if $i = j$ (we let $ID_i = ID^*$ at this point), $\mathcal{S}_2$ returns $upk_{ID_i} = Y$. Then $\mathcal{S}_2$ inserts a tuple $(ID_i, upk_{ID_i}, \perp)$ in a list L. Otherwise, $\mathcal{S}_2$ chooses $\nu_i \in \mathbb{Z}_q^*$ and sets $upk_{ID_i} = \nu_i P$ and $usk_{ID_i} = \nu_i$, and adds $(ID_i, upk_{ID_i}, usk_{ID_i})$ to L.

RevealSecretKey Oracle: Suppose the request is on an identity ID_i. If $ID_i \ne ID^*$, $\mathcal{S}_2$ finds $(ID_i, upk_{ID_i}, usk_{ID_i})$ in L and returns usk_{ID_i}. Otherwise, $\mathcal{S}_2$ halts the simulation.

Queries on Oracle H_0, H_0': Suppose ID_i is submitted to oracle $H_2(\cdot)$ and $H_2'(\cdot)$. $\mathcal{S}_2$ first scans $L_0 = \{(ID_i, Q_{ID_i}, Q'_{ID_i})\}$ to check whether H_0 and H_0' have already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{S}_2$ picks at random $Q_{ID_i}, Q'_{ID_i} \in \mathbb{G}_1$ and returns Q_{ID_i} and Q'_{ID_i} as the hash values of $H_0(ID_i)$ and $H_0'(ID_i)$ to $\mathcal{A}_2$, and also stores the values in the list L_0.

Queries on Oracle H_1: Suppose (m_i, ID_i, upk_{ID_i}) is submitted to oracle $H_1(\cdot)$. $\mathcal{S}_2$ first scans $L_1 = \{(m_i, ID_i, upk_{ID_i}, h_{i1})\}$ to check whether H_1 has already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{S}_2$ picks at random $h_{i1} \in \mathbb{Z}_q^*$ and returns h_{i1} as a hash value of $H_1(m_i, ID_i, upk_{ID_i})$ to $\mathcal{A}_2$ and also stores the values in the list L_1.

Queries on Oracle H_2, H_2': Suppose (m_i, ID_i, upk_{ID_i}) is submitted to oracle $H_2(\cdot)$ and $H_2'(\cdot)$. $\mathcal{S}_2$ first scans $L_2 = \{(m_i, ID_i, upk_{ID_i}, h_{i2}, h_{i2}')\}$ to check whether H_2 and H_2' have already been defined for that input. If so, the previously defined value is returned. Otherwise, $\mathcal{S}_2$ picks at random $h_{i2}, h_{i2}' \in \mathbb{Z}_q^*$ and returns h_{i2} and h_{i2}' as the hash values of $H_2(m_i, ID_i, upk_{ID_i})$ and $H_2'(m_i, ID_i, upk_{ID_i})$ to $\mathcal{A}_2$ and also stores the values in the list L_2.

Super-Sign Oracle: When $\mathcal{A}_2$ makes a **Super-Sign** query on m_i with ID_i, $\mathcal{S}_2$ first finds the corresponding $(ID_i, Q_{ID_i}, Q'_{ID_i})$ and $(ID_i, upk_{ID_i}, usk_{ID_i})$ from the lists L_0 and L, respectively. Then, $\mathcal{S}_2$ performs as follows:

- If $ID_i = ID^*$, $\mathcal{S}_2$ aborts the simulation.
- Otherwise, $\mathcal{S}_2$ picks three random $h_{i1}, h_{i2}, h_{i2}' \in \mathbb{Z}_q^*$ and $R_i \in \mathbb{G}_1$, and computes $\sigma_i = t' \cdot R_i + h_{i1} \cdot usk_{ID_i} \cdot X + h_{i2} s Q_{ID_i} + h_{i2}' s Q'_{ID_i}$. $\mathcal{S}_2$ then returns (R_i, σ_i) and adds $(m_i, ID_i, upk_{ID_i}, h_{i1})$ and $(m_i, ID_i, upk_{ID_i}, h_{i2}, h_{i2}')$ to L_1, L_2, respectively.

Eventually, $\mathcal{A}_2$ is to output a set of n users whith identities from the set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$ and corresponding public keys from the set $L_{upk}^* = \{upk_1^*, \cdots, upk_n^*\}$, n messages $L_m^* = \{m_1^*, \cdots, m_n^*\}$ and an aggregate signature (R^*, σ^*). It is required that there exist $ID^* \in \{ID_1^*, \cdots, ID_n^*\}$ such that $\mathcal{A}_2$ has not asked the secret key. Without loss of generality, we assume that $ID^* = ID_1^*$. And $\mathcal{A}_2$ has not made an (m_1^*, ID_1^*) query to **Super-Sign oracle**. Furthermore, the aggregate signature σ^* should satisfy the aggregate verification as follows:

$$\begin{aligned}\hat{e}(\sigma^*, P) &= \hat{e}(\sum_{i=1}^{n} h_{i1}^* upk_{ID_i^*}, Q)\hat{e}(R^*, Q') \\ &\quad \times \hat{e}(\sum_{i=1}^{n}(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), P_{pub}).\end{aligned} \tag{6}$$

Then, $\mathcal{S}_2$ finds the corresponding tuples (ID_i, upk_{ID_i}, usk_{ID_i}) for $2 \leq i \leq n$ from the list L. The following equation holds because the aggregate signature is valid:

$$\begin{aligned}\hat{e}(\sigma^*, P) &= \hat{e}(\sum_{i=1}^{n} h_{i1}^* upk_{ID_i^*}, X)\hat{e}(\sum_{i=1}^{n}(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), P_{pub})\hat{e}(R^*, Q') \\ &= \hat{e}(h_{11}^* Y, X)\hat{e}(\sum_{i=2}^{n} h_{i1}^* usk_{ID_i^*} P, X)\hat{e}(\sum_{i=1}^{n}(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), sP) \\ &\quad \times \hat{e}(R^*, t'P) \\ &= \hat{e}(h_{11}^* Y, X)\hat{e}(\sum_{i=2}^{n} h_{i1}^* usk_{ID_i^*} X, P)\hat{e}(\sum_{i=1}^{n} s(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}), P) \\ &\quad \times \hat{e}(t'R^*, P).\end{aligned} \tag{7}$$

Finally, $\mathcal{S}_2$ outputs abP as a solution to the CDH instance by computing

$$\begin{aligned}abP &= (h_{11}^*)^{-1}(\sigma^* - \sum_{i=2}^{n} h_{i1}^* usk_{ID_i^*} X - t'R^* \\ &\quad - \sum_{i=1}^{n} s(h_{i2}^* Q_{ID_i^*} + h_{i2}'^* Q'_{ID_i^*}))\end{aligned} \tag{8}$$

This completes the description of $\mathcal{S}_2$. It remains to show that $\mathcal{S}_2$ solves the given instance of the CDH problem with probability of at least ε'. To do so, we analyze three events needed for $\mathcal{S}_2$ to succeed:

- E_1 : $\mathcal{S}_2$ does not abort as a result of any of $\mathcal{A}_2$'s **RevealSecretKey** and **Super Sign** queries.
- E_2 : $\mathcal{A}_2$ generates a valid and non-trivial aggregate signature forgery.
- E_3 : Event E_2 occurs, $ID_1^* = ID^*$ and $ID_i^* \neq ID^*$ for all $i, 2 \leq i \leq n$.

$\mathcal{S}_2$ succeeds if all of these events happen. The probability $Pr[E_1 \wedge E_2 \wedge E_3]$ is decomposed as $Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid E_1 \wedge E_2]$.

Similarly to Lemma 1, it results that $\mathcal{S}_2$'s advantage in solving the CDH problem in $\mathbb{G}_1$ is at least $(1 - \frac{1}{q_{CreU}})^{q_{SupSig}+q_{RSec}+n-1} q_{CreU}\varepsilon$.

3.3. Comparison

We compare our scheme with existing CL-AS schemes (see Castro and Dahab, 2007; Gong et al., 2007; Zhang and Zhang, 2009; Zhang et al., 2010a). Regarding the computational overhead, we only consider the costly operations and we omit the computational efforts which can be delegated to pre-computing. We denote by P a pairing operation, by S a scalar multiplication in $\mathbb{G}_1$, by $\mathcal{P}_1$ the length of a point in $\mathbb{G}_1$. We know that pairing computation is more time-consuming than multiplication and exponentiation computation. From Table 2, we can see that in terms of computational cost our scheme is more efficient than the other schemes. As for the signature size, our signature requires two elements in $\mathbb{G}_1$ and approximately 320 bits.

Table 2. Comparison of CL-AS schemes

	size	Sign	Verify	Security	Syn.
Castro and Dahab (2007)	$\mathcal{P}_1$	$2S$	$(2n+1)P + nS$	Strong $\mathcal{A}$	×
Sch-I Sch-I, Gong et al. (2007)	$(n+1)\mathcal{P}_1$	$2S$	$(2n+1)P$	Strong $\mathcal{A}$	×
Sch-II, Gong et al. (2007)	$2\mathcal{P}_1$	$3S$	$(n+2)P + nS$	Strong $\mathcal{A}$	×
Zhang and Zhang (2009)	$(n+1)\mathcal{P}_1$	$3S$	$(n+3)P$	Super $\mathcal{A}$	✓
Zhang et al. (2010a)	$2\mathcal{P}_1$	$5S$	$5P + 2nS$	Strong $\mathcal{A}$	✓
Ours	$2\mathcal{P}_1$	$5S$	$4P + 3nS$	Super $\mathcal{A}$	×

On the other hand, only our scheme and the scheme in Zhang and Zhang (2009) are provably secure against the super Type I/II adversary in the random oracle model. However, their scheme needs synchronization, which cannot be easily achieved in the ad hoc networks.

4. Conclusion

We proposed an efficient CL-AS scheme requiring constant pairing operations and signature size, comparable to the most efficient CL-AS scheme so far. Our scheme has been proved secure against the super Type I/II adversary, which is the strongest attacker in the literatures. Our security proof is carried out under the standard CDH assumption. Furthermore, our scheme does not need synchronization. All these features distinguish our proposal from the existing ones as a practical scheme to secure mobile communication and computation applications.

Acknowledgements

This work has been partially supported by the Chinese NSF projects 61003230, 60970114, 60970115, 60970116, 61003214, 61173154, and 61021004, State Key

Laboratory of Information Security to Project 01-02-4, China Postdoctoral Science Foundation under Grant No. 20100480130. The second author is partly supported by the EU 7FP through project "DwB", the Spanish Government through projects CTV-09-634, PTA2009-2738-E, TSI-020302-2010-153, PT-430000- 2010-31, TIN2009-11689, CONSOLIDER INGENIO 2010 "ARES" CSD2007-0004 and TSI2007-65406-C03-01, by the Government of Catalonia under grant SGR2009-1135. The second author also acknowledges support by the Fundamental Research Funds for the Central Universities of China to Project 3103004, Beijing Municipal Natural Science Foundation to Project 4112052, and Shaanxi Provincial Education Department through Scientific Research Program 2010JK727. The authors would like to thank the chairs, committee, and reviewers of the security track of the INCoS 2011 conference.

5. References

AHN, J. H., GREEN, M. and HOHENBERGER, S. (2010) Synchronized Aggregate Signatures: New Definitions, Constructions and Applications. In: *Proceedings of the 17th ACM Computer and Communications Security (CCS 2010)*, Chicago, Illinois, USA. ACM Press, 473-484.

AL-RIYAMI, S.S. and PATERSON, K. (2003) Certificateless Public Key Cryptography. *Lecture Notes in Computer Science*, **2894**, 452-473.

BAGHERZANDI, A. and JARECKI, S. (2010) Identity-Based Aggregate and Multi-Signature Schemes Based on RSA. *Lecture Notes in Computer Science*, **6056**, 480-498.

BOLDYREVA, A., GENTRY, C., O'NEILL, A. and YUM, D. H. (2007)Ordered Multisignatures and Identity-Based Sequential Aggregate Signatures, with Applications to Secure Routing. In: *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007)*, Alexandria, Virginia, USA, 276-285.

BONEH, D., GENTRY, C., LYNN, B. and SHACHAM, H. (2003)Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *Lecture Notes in Computer Science*, **2656**, 416-432.

CASTRO, R. and DAHAB, R. (2007) Efficient Certificateless Signatures Suitable for Aggregation. [Online]. Retrieval Date: 1 December, 2011. Available from: http://eprint.iacr.org/2007/454.pdf

CHENG, X., LIU, J. and WANG, X. (2005)Identity-Based Aggregate and Verifiably Encrypted Signatures from Bilinear Pairing. *Lecture Notes in Computer Science*, **3483**, 1046-1054.

CHOI, K. Y., PARK, J. H., HUANG, J. Y. and LEE, D. H. (2007) Efficient Certificateless Signature Schemes. *Lecture Notes in Computer Science*, **4521**, 443-458.

CHOI, K. Y., PARK, J. H. and LEE, D. H. (2011) A New Provably Secure Certificateless Short Signature Scheme. *Computers and Mathematics with Applications*, **61**(7), 1760-1768.

European Parliament (2005) Legislative resolution on the proposal for a directive of the European Parliament and of the Council on the retention of data processed in connection with the provision of public electronic communication services and amending Directive 2002/58/EC (COM(2005)0438 C6-0293/2005 2005/0182(COD)).

Gentry, C. and Ramzan, Z. (2006) Identity-Based Aggregate Signatures. *Lecture Notes in Computer Science*, **3958**, 257-273.

Gong, Z., Long, Y., Hong, X. and Chen, K. (2007) Two Certificateless Aggregate Signatures from Bilinear Maps. In: *Proceedings of the 8th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, Qingdao, China. IEEE, 188-193.

Gorantla, M.C. and Saxena, A. (2005) An Efficient Certificateless Signature Scheme. *Lecture Notes in Computer Science*, **3802**, 110-116.

Herranz, J. (2006) Deterministic Identity-Based Signatures for Partial Aggregation. *The Computer Journal*, **49**(3), 322-330.

Hu, B.C., Wong, D.S., Zhang, Z. and Deng, X. (2006) Key Replacement Attack Against a Generic Construction of Certificateless Signature. *Lecture Notes in Computer Science*, **4058, 235-246.**

Huang, X.Y., Susilo, W., Mu Y. and Zhang, F. (2005) On the Security of Certificateless Signature Schemes from Asiacrypt 2003. *Lecture Notes in Computer Science*, **3810**, 13-25.

Huang, X., Mu, Y., Susilo, W. Wong, D. S. and Wu, W. (2007) Certificateless Signature Revisited. *Lecture Notes in Computer Science*, **4586**, 308-322.

Lu, S., Ostrovsky, R., Sahai, A., Shacham, H. and Waters, B. (2006) Sequential Aggregate Signatures and Multisignatures Without Random Oracles. *Lecture Notes in Computer Science*, **4004**, 465-485.

Lysyanskaya, A., Micali, S., Reyzin, L. and Shacham, H. (2004) Sequential Aggregate Signatures from Trapdoor Permutations, *Lecture Notes in Computer Science*, **3027**, 74-90.

Neven, G. (2008) Efficient Sequential Aggregate Signed Data. *Lecture Notes in Computer Science*, **4965**, 52-69.

Shamir, A. (1984) Identity-Based Cryptosystems and Signature Schemes. *Lecture Notes in Computer Science*, **196**, 47-53.

Shim, K. A. (2010) An ID-Based Aggregate Signature Scheme with Constant Pairing Computations. *The Journal of Systems and Software*, **83**(10), 1873-1880.

Wu, Q., Domingo-Ferrer, J. and Gonzalez-Nicolas, U. (2010) Balanced Trustworthiness, Safety, and Privacy in Vehicle-to-Vehicle Communications. *IEEE Transactions on Vehicular Technology*, **59**(2), 559-573.

Wu, Q., Mu, Y., Susilo, W., Qin, B. and Domingo-Ferrer, J. (2009) Asymmetric Group Key Agreement. *Lecture Notes in Computer Science*, **5479**, 153–170.

WU, Q., QIN, B., ZHANG, L., DOMINGO-FERRER, J., and FARRÀS, O. (2011) Bridging Broadcast Encryption and Group Key Agreement. *Lecture Notes in Computer Science*, **7073**, 143-160.

XIONG, H., WU, Q. and CHEN, Z. (2011) Strong Security Enabled Certificateless Aggregate Signatures Applicable to Mobile Computation. In: *Proceedings of the 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, Fukuoka, Japan. IEEE, 92-99.

XU, J., ZHANG, Z. and FENG, D. (2005) ID-based Aggregate Signatures from Bilinear Pairings, *Lecture Notes in Computer Science*, **3810**, 110-119.

YUM, D. H. and LEE, P. J. (2004) Generic Construction of Certificateless Signature. *Lecture Notes in Computer Science*, **3108**, 200-211.

ZHANG, Z. and WONG, D. (2006) Certificateless Public-Key Signature: Security Model and Efficient Construction. *Lecture Notes in Computer Science*, **3989**, 293-308.

ZHANG, L. and ZHANG. F. (2008) A New Provably Secure Certificateless Signature Scheme. *Proceedings of the International Conference on Communications (ICC 2008)*, Beijing, China. IEEE, 1685-1689.

ZHANG, L. and ZHANG, F. (2009) A New Certificateless Aggregate Signature Scheme. *Computer Communications*, **32**(6), 1079-1085.

ZHANG, L., QIN, B., WU, Q. and ZHANG, F. (2010) Efficient Many-to-One Authentication with Certificateless Aggregate Signatures. *Computer Networks*, **54**(14), 2482-2491.

ZHANG, L., WU, Q., SOLANAS, A. and DOMINGO-FERRER, J. (2010) A Scalable Robust Authentication Protocol for Secure Vehicular Communications. *IEEE Transactions on Vehicular Technology*, **59**(4), 1606-1617.