Journal of King Saud University – Computer and Information Sciences (2014) **26**, 276–286

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Certificateless aggregate signcryption: Security model and a concrete construction secure in the random oracle model

Ziba Eslami *, **Nasrollah Pakniat**

Department of Computer Sciences, Shahid Beheshti University, G.C., Tehran, Iran

Received 22 October 2012; revised 3 November 2013; accepted 13 March 2014
Available online 9 May 2014

KEYWORDS

Certificateless cryptography;
Aggregate signcryption;
Random oracle model;
Bilinear pairing

Abstract The concept of aggregate signcryption was first introduced in 2009 by Selvi et al. [Identity based aggregate signcryption schemes, Lecture Notes in Computer Science 5922 LNCS, 2009, pp. 378–397]. The aggregation process of these schemes reduces the amount of exchanged information and is particularly useful in low-bandwidth communication networks and computationally-restricted environments such as wireless sensor networks. Selvi et al.'s scheme is in the identity-based setting and suffers from the key escrow problem. The goal of this paper is to overcome this problem and propose a suitable security model for aggregate signcryption in the certificateless setting. We further propose a concrete certificateless aggregate signcryption scheme which is based on Barbosa and Farshim's certificateless signcryption scheme [Certificateless signcryption. In: M. Abe, V. Gligor (Eds.), Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS-08), ACM, New York. pp. 369–372]. We then prove the security of the proposed scheme in the random oracle model under the gap Bilinear Diffie–Hellman and computational Diffie–Hellman intractability assumptions.

1. Introduction

Among important cryptographic primitives, one can name encryption and signature schemes. Signcryption is another cryptographic primitive that simultaneously achieves the security objectives of both encryption and signatures and has a lower computational cost and communication overhead than the sign-then-encrypt approach. Consider a situation in which a set of n distinct users $\{u_i\}_{i=1}^{n}$ wants to send messages $\{m_i\}_{i=1}^{n}$ to a designated recipient (R). The signcryption primitive can be used to provide simultaneously confidentiality for the senders $\{u_i\}_{i=1}^{n}$ and authenticity for the receiver$R$. In this scenario, the messages are signcrypted as $\{c_i\}_{i=1}^{n}$ and then sent to R. The recipient first verifies if the received ciphertexts are signcrypted by $\{u_i\}_{i=1}^{n}$ and then decrypts them to obtain $\{m_i\}_{i=1}^{n}$. Therefore, it is desirable to shorten the amount of exchanged information and to reduce the computational complexity of the verification process. These are particularly crucial in low-bandwidth communication networks or computationally

* Corresponding author. Tel.: +98 21 22903007; fax: +98 21 22431655.
E-mail address: z_eslami@sbu.ac.ir (Z. Eslami).

Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

1319-1578 © 2014 Production and hosting by Elsevier B.V. on behalf of King Saud University.
http://dx.doi.org/10.1016/j.jksuci.2014.03.006

restricted environments. An aggregate signcryption scheme is an algorithm to efficiently aggregate $\{c_i\}_{i=1}^n$ into a single ciphertext c in such a way that the recipient R is able to do the following:

(1) verify that c is signcrypted by $\{u_i\}_{i=1}^n$ and
(2) extract $\{m_i\}_{i=1}^n$ from c.

Aggregate signcryption schemes can be applied in services such as online polling, online banking, e-auction, routing scenarios, traffic management systems, etc. As an example, we demonstrate how we can apply an aggregate signcryption scheme to improve an e-auction system:

Consider an e-auction event. The bidders want to ensure that their proposals are hidden, and only the contracting authority will be able to view their proposals. The contracting authority would also want to ensure that this is a valid proposal by a valid bidder. In situations like this, we can employ the signcryption primitive, which provides both confidentiality for the senders and authentication to the receiver. The contracting authority will be a secure device, but the computational power of the device might be limited. Providing high security and high computational power results in a huge cost demand. The security of such devices cannot be relaxed, and thus, by reducing the computation power, one can greatly reduce the cost involved. Hence, with limited computational power, a contracting authority will find it very difficult to verify the authenticity of each and every proposal separately (as it is possible for malicious bidders to contribute to bidding). By using aggregate signcryption, the contracting authority will easily be able to verify the authenticity of all of the proposals using a single verification step. As one part of all of the signcryptions is aggregated, the bandwidth is also saved. Thus, aggregate signcryption can play a very important role in this scenario. After verifying the proposals, the contracting authority outputs the bidder with the minimum proposal as the winner.

In this paper, we are mainly concerned with the certificateless setting and propose a suitable security model for certificateless aggregate signcryption schemes (CLASC). We outline in detail the framework of an aggregate signcryption scheme in the certificateless public key setting. We further propose a CLASC scheme, which we prove to be secure in the random oracle model.

The rest of the paper is organized as follows. In Section 2, related works is summarized. Section 3 introduces preliminary material. In Section 4, we introduce the formal definition of the security model of CLASC schemes. In Section 5, we describe the proposed CLASC scheme, and in Section 6, we analyze it. Finally, conclusions are provided in Section 7.

2. Related works

Certificateless cryptography, put forward first by Al-riyami and Paterson (2003), can be considered as an intermediate solution to overcome the issues in traditional public key infrastructure (PKI) and identity-based public key cryptography (ID-PKC). Whereas a trusted authority is needed in traditional PKI to bind the identity of an entity to his public key, ID-PKC, introduced by Shamir (1984), requires a trusted private key generator (PKG) to generate the private keys of users based on their identities. Therefore, the certificate management problem in the public-key setting is actually replaced by the key escrow problem.

In certificateless public key cryptography, we still employ a third party, called the key generation center (*KGC*), to help users generate their private keys. However, the*KGC*does not have access to the final private keys that are generated by the users themselves (based on the partial private keys obtained from the *KGC* and the secret information chosen by the users). The public key of a user is computed from the *KGC*'s public parameters and some information, private to the user, and is published by the user himself.

There exists a vast number of encryption and digital signature schemes in certificateless cryptography. For more details about such schemes, we refer the interested reader to Baek et al. (2005), Hu et al. (2007), Long and Chen (2007), Zhang and Zhang (2008), Duan (2008), Guoyan and Xiaoyun (2009), Chang et al. (2009), Sun and Zhang (2010), Zhao and Ye (2012), Tso et al. (2012), Seo et al. (2012), Zhang and Mao (2012), Islam and Biswas (2013a,b).

Zheng (1997) proposed the concept of signcryption, which has a lower computational cost and communication overhead than the sign-then-encrypt approach. Zheng further proposed a concrete signcryption scheme based on the discrete logarithm problem. Identity-based signcryption schemes and the definitions of their security model dealing with the notions of privacy and unforgeability are considered in Malone-Lee (2002), Libert and Quisquater (2003), Boyen (2003) and Chen and Malone-Lee (2005).

The first certificateless signcryption (CLSC) scheme is proposed by Barbosa and Farshim (2008). The authors claimed that their scheme is secure in the random oracle model. Selvi et al. (2009a) demonstrated that Barbosa and Farshim's scheme is existentially forgeable. Liu et al. (2010) proposed another CLSC scheme and claimed that it is secure in the standard model. However, Weng et al. (2011) and Miao et al. (2013) demonstrated that Liu et. al.'s scheme is completely insecure.

The concept of an aggregate signcryption scheme was introduced in Selvi et al. (2009b). The authors also defined a suitable security model for identity-based aggregate signcryption schemes and proposed examples that are proved to be secure in the random oracle model. We also note that aggregate signature schemes have been discussed in the literature for a while. The interested readers can find more such materials in Boneh et al. (2003), Lysyanskaya et al. (2004), Xu et al. (2005), Cheng et al. (2006), Gentry and Ramzan (2006), Zhang and Zhang (2009), Zhang et al. (2010) and Xiong et al. (2013).

3. Preliminaries

In this section, we provide a brief review of preliminary material including bilinear maps and some mathematical problems.

Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same order. A map $e{:}G_1 \times G_1 \rightarrow G_2$ is called a bilinear map if it satisfies the following properties:

(1) Bilinearity: $e(aP,bQ) = e(P,Q)^{ab}$ for all $P,Q \in G_1$ and $a,b \in Z_q^*$.

(2) Non-degeneracy: There exists $P,Q \in G_1$ such that $e(P,Q) \neq 1$.
(3) Computability: There exists an efficient algorithm to compute $e(P,Q)$ for any$P,Q \in G_1$.

Computational Diffie–Hellman (*CDH*) Problem: Given a generator P of an additive group G with order$q$ and (aP,bP) for unknown $a,b \in Z_q^*$, compute abP.

Decisional Bilinear Diffie–Hellman (*DBDH*) Problem: Given a generator P of an additive group G with order$q$and (aP,bP,cP,t) for unknown $a,b,c \in Z_q^*$, decide $e(P,P)^{abc} = t$ or not.

Gap Bilinear Diffie–Hellman (*GBDH*) Problem: Given $(P,aP,bP,cP \in G_1)$ for some unknown $a,b,c \in Z_q^*$, compute $w = e(P,P)^{abc} \in G_2$ with the help of *DBDH* oracleO_{DBDH}.

Gap Diffie–Hellman (*GDH'*) Problem: Given $(P,aP,bP \in G_1)$ for some unknown $a,b \in Z_q^*$, compute $abP \in G_1$ with the help of *DBDH* oracleO_{DBDH}.

4. Certificateless aggregate signcryption schemes

In this section, we first define the framework of a certificateless aggregate signcryption scheme. We then propose a formal definition for the security model of CLASC schemes.

4.1. Framework of CLASC schemes

The participants involved in a certificateless aggregate signcryption scheme consist of a key generation center (*KGC*), an aggregating set u of n users $(\{u_i\}_{i=1}^n)$, a receiver u_R and an aggregate signcryption generator. The probabilistic polynomial-time (PPT) algorithms: Setup, PartialPrivateKeyExtract, UserKeyGenerate, Signcrypt, Aggregate, AggregateVerify and AggregateUnsigncrypt should also be defined. The description of each algorithm is as follows:

Setup: This algorithm takes as input a security parameter l and returns *params* (system parameters) and a randomly chosen master secret key *msk*. The *KGC* carries out the algorithm and publishes *params*. *msk* will be kept secret.

PartialPrivateKeyExtract: This algorithm takes as input *params*, *msk* and an identity $ID_u \in \{0,1\}^*$ of an entity u, and returns a partial private key D_u. The *KGC* carries out the algorithm to generate the partial private key D_u and sends D_u to the corresponding owner u via a secure channel.

UserKeyGenerate: This algorithm takes *params* and an entity's identity ID_u as input and returns a randomly chosen secret value x_u and a corresponding public key P_u for the entity. The entity u runs the algorithm to generate his public key and then distributes the public key P_u without being certified.

Signcrypt: An algorithm run by each user u_i in an aggregating set$u$. u_i's inputs are the system parameters *params*, some state information Δ, a message M_i, his identity ID_i, his corresponding public key P_i, his private key (x_i,D_i), the identity of the receiver ID_R and his corresponding public key P_R. This algorithm outputs a ciphertext c_i. This is a probabilistic algorithm.

Aggregate: An algorithm run by the aggregate signcryption generator that takes as inputs an aggregating set u of n users $(\{u_i\}_{i=1}^n)$, some state information Δ, the identity ID_i of each sender u_i, the corresponding public key P_i of each u_i, and a ciphertext c_i on a message M_i under the identity ID_i and public key P_i for each user $u_i \in u$ ciphered with the state information Δ to a user with identity ID_R and the corresponding public key P_R. The output of this algorithm is an aggregate ciphertext c on messages $\{M_i\}_{i=1}^n$, where the aggregate signcryption generator does not know M_is.

AggregateVerify: This algorithm takes as input an aggregating set u of n users $\{u_i\}_{i=1}^n$, the identity ID_i and the corresponding public key P_i of each user u_i, the identity of the receiver ID_R and his corresponding public key P_R, the state information Δ and an aggregate ciphertext c. It outputs true if the aggregate signcryption is valid or false otherwise.

AggregateUnsigncrypt: This algorithm takes an aggregate ciphertext c, the state information Δ, the receiver's full private key (x_R,D_R), his identity ID_R and his public key P_R, and the senders' identities $\{ID_i\}_{i=1}^n$ and corresponding public keys $\{P_i\}_{i=1}^n$ as input and outputs a set of n plaintexts $\{M_i\}_{i=1}^n$. Typically, the AggregateUnsigncrypt algorithm is a deterministic algorithm.

As in Gentry and Ramzan (2006), Zhang and Zhang (2009) and Zhang et al. (2010), in an aggregating set, all of the users must use the same (unique) state information Δ in the Signcrypt algorithm. For such a Δ, one can choose the current time, some part of the system parameters or other feasible information.

4.2. The proposed security model for certificateless aggregate signcryption schemes

The security model for certificateless signcryption schemes is introduced by Barbosa and Farshim (2008). In this section, we propose a security model for certificateless aggregate signcryption schemes. The ciphertext indistinguishability and the existential unforgeability security models are used to capture the confidentiality and authenticity requirements, respectively. As for the adversarial model, we follow the common approach in the certificateless setting, which considers two types of adversaries. A Type *I* adversary A_I who does not have access to the master secret key but can replace the public key of any entity with another value and a Type *II* Adversary A_{II} who has access to the master secret key but is unable to perform public key replacement. We now define the required security games to capture.

4.2.1. Confidentiality requirement

The confidentiality property is defined based on the concept of indistinguishability of encryptions under adaptively chosen ciphertext attacks (IND-CCA2). We define the following two games against Type *I* and Type *II* adversaries.

Game I. The game is performed by a challenger $\mathcal{C}$ and a Type *I* adversary A_I.

- *Initialization.* $\mathcal{C}$ runs the Setup algorithm to generate a master secret key *msk* and the public system parameters *params*. $\mathcal{C}$ keeps *msk* secret and gives *params* to A_I. Note that A_I does not know *msk*.
- *Phase 1.* A polynomially bounded number of the following queries is performed by A_I. The queries can be made adaptively so that answers to the previous queries might affect subsequent ones.

- *RequestPublicKey*. When A_I supplies an identity ID_u and requests u's public key, $\mathcal{C}$ responds with the public key P_u for the identity.
- *ExtractPartialPrivateKey*. When A_I supplies an identity ID_u and requests u's partial private key, $\mathcal{C}$ responds with the partial private key D_u for the identity.
- *ReplacePublicKey*. When A_I supplies an identity ID_u and a new valid public key value P'_u, $\mathcal{C}$ replaces the current public key value with the value P'_u.
- *ExtractSecretValue*. When A_I requests the secret value of an identity ID_u, the challenger returns the secret value x_u of u. The public key of u should not have been replaced byA_I.
- *Signcrypt*. When A_I submits a sender with an identity ID_S, a receiver with an identity ID_R, a message M and some state information Δ to the challenger, $\mathcal{C}$ responds by running the Signcrypt algorithm on the message M, the state information Δ, the sender's private key (D_S,x_S) and the receiver's public key P_R.
- *AggregateUnsigncrypt*. When A_I submits an aggregate ciphertext c, some state information Δ, senders with identities $\{ID_i\}_{i=1}^n$ and a receiver with the identity ID_R, $\mathcal{C}$ checks the validity of c and if it is a valid ciphertext, then $\mathcal{C}$ returns the result of running the AggregateUnsigncrypt algorithm on the ciphertext c, the state information Δ, the receiver's private key (D_R,x_R) and the senders' public keys $\{P_i\}_{i=1}^n$.

- *Challenge*. When Phase *1* ends, the adversary outputs $n+1$ distinct identities $\{ID_i^*\}_{i=1}^n$, ID_R^*, some state information Δ^* and two sets of n messages $M_0^* = \{m_{0i}^*\}_{i=1}^n$, $M_1^* = \{m_{1i}^*\}_{i=1}^n$. Now, a bit μ is randomly chosen by $\mathcal{C}$ who then produces c^* as the aggregate signcryption of messages M_μ^* using the state information Δ^*, the private keys corresponding to $\{ID_i^*\}_{i=1}^n$ and the public key and the identity of u_R^*. The challenger returns c^* to the adversary.
- *Phase 2*. The adversary can continue to probe the challenger as in Phase 1.
- *Response*. The adversary returns a bit μ'.

We say that the adversary wins the game if $\mu' = \mu$, subject to the following conditions:

(1) A_I never queries the partial private key for ID_R^*.
(2) A_I cannot make an AggregateUnsigncrypt query on c^* under ID_R^* and $\{ID_i'\}_{i=1}^n$ where at least for one $i, ID_i^* = ID_i'$. The only exception is when the public key P_i^* of all of the senders ID_j^* with $ID_j^* = ID_j'$ or that of the receiver P_R^* used to signcrypt M_μ^* have been replaced after the challenge was issued.

The advantage of A_I is defined as follows:

$$Adv_{A_I}^{IND-CLASC-CCA2} = |2Pr[\mu = \mu'] - 1|,$$

where $Pr[\mu = \mu']$ denotes the probability that $\mu = \mu'$.

Game II. The game is performed by a challenger $\mathcal{C}$ and a Type *II* adversary A_{II}.

- *Initialization*. $\mathcal{C}$ first generates (*params*, *msk*) and outputs them to A_{II}.
- *Phase 1*. A_{II} may adaptively make a polynomially bounded number of queries as in Game *I*. The only constraint is that A_{II} cannot replace any public keys. Note that since A_{II} knows the master secret key, it can compute the partial private key of any identity.
- *Challenge*. When Phase *1* ends, the adversary outputs $n+1$ distinct identities $\{ID_i^*\}_{i=1}^n$, ID_R^*, some state information Δ^* and two sets of n messages $M_0^* = \{m_{0i}^*\}_{i=1}^n$, $M_1^* = \{m_{1i}^*\}_{i=1}^n$. Now, a bit μ is randomly chosen by $\mathcal{C}$ who then produces c^* as the aggregate signcryption of messages M_μ^* using the state information Δ^*, the private keys corresponding to $\{ID_i^*\}_{i=1}^n$ and the public key of u_R^*. The challenger returns c^* to the adversary.
- *Phase 2*. The adversary can continue to probe the challenger as in Phase 1.
- *Response*. The adversary returns a bit μ'.

We say that the adversary wins the game if $\mu = \mu'$, and the following constraints are fulfilled:

(1) A_{II} never queries the secret value for the challenge identity ID_R^*.
(2) In Phase *2*, A_{II} cannot make an AggregateUnsigncrypt query for the challenge ciphertext c^* under ID_R^* and $\{ID_i'\}_{i=1}^n$, where at least for one i, $ID_i^* = ID_i'$.

As in Game *I*, the advantage of A_{II} is defined as follows:

$$Adv_{A_{II}}^{IND-CLASC-CCA2} = |2Pr[\mu = \mu'] - 1|.$$

Definition 1. A CLASC scheme is semantically secure under adaptively chosen ciphertext attack if no PPT adversary (of either Type) has a non-negligible advantage in Game *I* or Game *II*.

Note that as the adversaries can access the private keys of all of the senders, the above definition of security assures that confidentiality is preserved even if these keys are compromised, and, therefore, insider security is guaranteed.

4.2.2. Authenticity requirement

The authenticity property is defined based on existential unforgeability against chosen message attack (EUF-CMA) and is captured by the following two games against Type *I* and Type *II* adversaries.

Game III. The game is performed by a challenger $\mathcal{C}$ and a Type *I* adversary A_I.

- *Setup*. $\mathcal{C}$ runs the Setup algorithm and takes as input a security parameter to obtain a master secret key and the system parameters *params*. $\mathcal{C}$ then sends *params* to the adversary A_I while keeping the master secret key secret.
- *Phase 1*. A_I may adaptively make a polynomially bounded number of queries as in Game *I*.
- *Forgery*. A_I outputs $n+1$ users $\{u_i^*\}_{i=1}^n$ and u_R^*, with identities $\{ID_i^*\}_{i=1}^n$ and ID_R^* and corresponding public keys $\{P_i^*\}_{i=1}^n$ and P_R^*, n messages $\{M_i^*\}_{i=1}^n$, some state information Δ^* and an aggregate ciphertext c^*.

We say that A_I wins Game *III* if the following holds:

(1) c^* *is a valid aggregate ciphertext* associated with the state information Δ^*, senders' identities $\{ID_i^*\}_{i=1}^n$ and corresponding public keys $\{P_i^*\}_{i=1}^n$, the receiver's identity ID_R^* and the corresponding public key P_R^*.
(2) The partial private key of at least one of the members of the aggregating set, say ID_1^*, has not been queried during the ExtractPartialPrivateKey queries. We further require that signcryption of $(M_1^*, \Delta^*, ID_1^*, ID_R^*)$ has never been queried during the Signcrypt queries.

Game IV. The game is performed by a challenger $\mathcal{C}$ and a Type *II* adversary A_{II}.

- *Setup*. $\mathcal{C}$ runs the Setup algorithm to generate *params* and *msk* and then sends them to A_{II} as in Game *II*.
- *Phase 1*. A_{II}'s queries are identical to those of Game *II*.
- *Forgery*. A_{II} outputs $n + 1$ users $\{u_i^*\}_{i=1}^n$ and u_R^*, with identities $\{ID_i^*\}_{i=1}^n$ and ID_R^*, and corresponding public keys $\{P_i^*\}_{i=1}^n$ and P_R^*, n messages $\{M_i^*\}_{i=1}^n$, some state information Δ^* and an aggregate ciphertext c^*.

We say that A_{II} wins Game *IV*, iff

(1) c^* *is a valid aggregate ciphertext* associated with the state information Δ^*, senders' identities $\{ID_i^*\}_{i=1}^n$ and the corresponding public keys $\{P_i^*\}_{i=1}^n$, the receiver's identity ID_R^* and the corresponding public key P_R^*.
(2) One of the identities, without loss of generality, say ID_1^* in the set of signcrypters has not been submitted during the ExtractSecretValue queries and signcryption of $(M_1^*, \Delta^*, ID_1^*, ID_R^*)$ has never been queried during the Signcrypt queries.

Definition 2. A CLASC scheme is existentially unforgeable under adaptively chosen message attacks if no PPT adversary (of either Type) has a non-negligible advantage in the Game *III* or the Game *IV*.

Note that as the adversaries can access the private key of the receiver, the above definition of security assures that unforgeability is preserved even if this key is compromised, and, therefore, insider security is guaranteed.

5. The proposed certificateless aggregate signcryption scheme

In this section, we present a concrete CLASC scheme that is based on the scheme of Barbosa and Farshim (2008), which uses the Encrypt-then-Sign approach with shared randomness and public verifiability of the ciphertext. In Selvi et al. (2009a), it is demonstrated that since the scheme of Barbosa and Farshim (2008) does not bind the receiver's identity to the signature, then it is existentially forgeable. Therefore, we have to make some modifications to address this problem. The details of the proposed scheme are as follows.

- *Setup:* performed by *KGC*.
  - Input: the security parameter l.
  - Process:
    (1) choose a cyclic additive group G_1 generated by P of prime order q,
    (2) choose a cyclic multiplicative group G_2 of the same order and a bilinear map $e: G_1 \times G_1 \to G_2$,
    (3) choose a random number $s \in Z_q^*$ as the master secret key and set $Mpk = sP$,
    (4) choose cryptographic hash functions $H_1: \{0,1\}^* \to G_1, H_2: \{0,1\}^* \to \{0,1\}^k$, $H_3: \{0,1\}^* \to G_1, H_4: \{0,1\}^* \to G_1$.
  - Output: the master secret key s, which will be secured by *KGC*and the system parameters $params = (G_1, G_2, e, P, Mpk, H_1, H_2, H_3, H_4)$, which is published.

- *PartialPrivateKeyExtract:* performed by *KGC*.
  - Input: *params*, master secret key s and a user's identity $ID_i \in \{0,1\}^*$.
  - Process:
    (1) compute $Q_i = H_1(ID_i)$,
    (2) compute $D_i = sQ_i$.
  - Output: the partial private key D_i, which is sent securely to the user with identity ID_i.

- *UserKeyGenerate:* performed by each user of the system.
  - Input: user's identity ID_i.
  - Process:
    (1) select a random number $x_i \in Z_q^*$,
    (2) compute $P_i = x_i P$ as the user's public key.
  - Output: x_i and P_i, of which the first one will be secured by this user, and the second one is published.

- *Signcrypt:* run by user u_i.
  - Input: u_i's identity ID_i and his public/private key, the receiver's identity ID_R and his public key, some state information Δ and a message M_i.
  - Process:
    (1) choose a random number $r_i \in Z_q^*$ and compute $U_i = r_i P$, $T_i = e(Mpk, Q_R)^{r_i}$,
    (2) compute $h_i = H_2(U_i, T_i, r_i P_R, ID_R, P_R, \Delta)$,
    (3) compute $V_i = h_i \oplus M_i$,
    (4) compute $H_i = H_3(U_i, V_i, ID_i, P_i, ID_R, P_R)$,
    (5) compute $H' = H_4(\Delta)$,
    (6) compute $W_i = D_i + r_i H_i + x_i H'$.
  - Output: $c_i = (U_i, V_i, W_i)$.

- *Aggregate:* performed by aggregate signcryption generator.
  - Input: a collection of individual ciphertexts $\{c_i = (U_i, V_i, W_i)\}_{i=1}^n$ generated by users with identity $\{ID_i\}_{i=1}^n$ to a receiver with identity ID_R under the same state information Δ.
  - Process: compute $W = \sum_{i=1}^n W_i$.
  - Output: aggregate ciphertext $c = (U_1, \ldots, U_n, V_1, \ldots, V_n, W)$.

- *AggregateVerify:* performed by an arbitrary entity.
  - Input: an aggregate ciphertext $c = (U_1, \ldots, U_n, V_1, \ldots, V_n, W)$ generated by n users $\{u_i\}_{i=1}^n$ with identities $\{ID_i\}_{i=1}^n$ and corresponding public keys $\{P_i\}_{i=1}^n$ for a receiver u_R with identity ID_R and the corresponding public key P_R using the same state information Δ.

- Process:
  (1) compute $H_i = H_3(U_i,V_i,ID_i,P_i,ID_R,P_R)$, For $i = 1,\ldots,n$,
  (2) compute $H' = H_4(\Delta)$,
  (3) verify $e(W,P) =^{?} e\left(\sum_{i=1}^{n} Q_i, Mpk\right)\prod_{i=1}^{n} e(H_i,U_i)\, e\left(H', \sum_{i=1}^{n} P_i\right)$.
- Output: true if the above equation holds, false otherwise.

- *AggregateUnsigncrypt:* performed by the receiver (if the output of AggregateVerify algorithm is true).
  - Input: the inputs of AggregateVerify algorithm and the private key of the receiver (x_R,D_R).
  - Process: for $i = 1,\ldots,n$:
    (1) compute $T_i = e(U_i,D_R)$,
    (2) compute $h_i = H_2(U_i,T_i,x_RU_i,ID_R,P_R,\Delta)$,
    (3) compute $M_i = V_i \oplus h_i$.
  - Output: $\{M_i\}_{i=1}^{n}$.

Using the technique described in Al-riyami and Paterson (2003), our scheme can easily achieve trust level 3.

6. Analysis of the proposed scheme

6.1. Correctness

The following equalities show the correctness of the verification algorithm:

$$\begin{aligned} e(W,P) &= e\left(\sum_{i=1}^{n} W_i, P\right) \\ &= e\left(\sum_{i=1}^{n} (D_i + r_iH_i + x_iH'), P\right) \\ &= e\left(\sum_{i=1}^{n} (sQ_i + r_iH_i + x_iH'), P\right) \\ &= e\left(\sum_{i=1}^{n} Q_i, Mpk\right)\prod_{i=1}^{n} e(H_i,U_i)\, e\left(H', \sum_{i=1}^{n} P_i\right). \end{aligned}$$

6.2. Security analysis

Assuming that the *CDH*, *GDH′* and *GBDH* problems are hard, we now demonstrate the security of our CLASC scheme.

Theorem 1. *The certificateless aggregate signcryption scheme of Section 5 is indistinguishable against adaptive chosen ciphertext attack in the random oracle model.*

This theorem follows from Lemmas 1 and 2.

Lemma 1. *Assuming the intractability of GBDH problem, the proposed CLASC scheme is secure against adversary A_I during Game I under adaptive chosen ciphertext attack in the random oracle model.*

Proof. We suppose that a Type *I* adversary A_I exists for our scheme. We now demonstrate how to use A_I to devise an algorithm $\mathcal{C}$ that can solve the *GBDH* problem. □

We provide $\mathcal{C}$ with the *GBDH* challenge (P,aP,bP,cP) as input. $\mathcal{C}$ sets $Mpk = aP$ and sends G_1,G_2,e,P and Mpk to A_I. $\mathcal{C}$ chooses a random number $l \leqslant q_{H_1}$ as the challenge identity, where q_{H_1} is the maximum number of queries that A_I could place to H_1 oracle. To solve the *GBDH* problem, $\mathcal{C}$ replies to queries made by A_I. In the following, we describe how $\mathcal{C}$ responds to these queries. Note that to avoid collision and consistently respond to these queries, $\mathcal{C}$ has to maintain some lists, which are all initially empty (this assumption holds throughout the rest of the paper).

H_1 *queries:* On the *i*-th (non-repeated) query *ID*, if $i \neq l$, then $\mathcal{C}$ chooses $r \in Z_q^*$ uniformly at random and sets $Q_{ID} = rP$. It then adds (i,ID,r) to a list L_1 and returns Q_{ID}. Otherwise, it returns $Q_{ID} = bP$ and adds $(l,ID,\perp)$ to L_1, where $\perp$ is an empty string. We denote by ID_l, the *l*-th non-repeated identity queried to this oracle.

ExtractPartialPrivateKey queries: For each new query *ID*, $\mathcal{C}$ inputs *ID* to H_1 and obtains (i,ID,r). If $i = l$, then $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ returns $D = raP$.

ExtractSecretValue queries: On input *ID*, $\mathcal{C}$ searches L_K for the entry (ID,PK,x) corresponding to *ID*. If no entry is found, $\mathcal{C}$ generates the key pair (PK,x) and adds the tuple (ID,PK,x) to the list. In both cases, $\mathcal{C}$ returns*x*.

RequestPublicKey queries: On input *ID*, $\mathcal{C}$ searches L_K for the entry (ID,PK,x) corresponding to *ID*. If no entry is found, $\mathcal{C}$ generates the key pair (PK,x) and adds the tuple (ID,PK,x) to the list. $\mathcal{C}$ then returns *PK*.

ReplacePublicKey queries: On input (ID,PK), $\mathcal{C}$ inserts/updates L_K with the tuple $(ID,PK,\perp)$.

H_3 *queries:* On input (U,V,ID,PK,ID',PK'), $\mathcal{C}$ searches L_3 for the entry (U,V,ID,PK,ID',PK',t,tP). If no entry is found, $\mathcal{C}$ generates a random value *t* in Z_q^*. It then inserts the tuple (U,V,ID,PK,ID',PK',t,tP) in the list L_3. In both cases, $\mathcal{C}$ returns *tP*.

H_4 *queries:* On input (Δ), $\mathcal{C}$ searches L_4 for the entry (Δ,s,sP). If no entry is found, $\mathcal{C}$ generates a random value *s* in Z_q^*. It then inserts the tuple (Δ,s,sP) in the list L_4. In both cases, $\mathcal{C}$ returns *sP*.

H_2 *queries:* On input (U,T,R,ID,PK,Δ), $\mathcal{C}$ proceeds as follows:

(1) C checks if the decision bilinear Diffie–Hellman oracle returns *1* when queried with one of the tuples $\{aP, bP, d_icP, T\}_{i=1}^{n}$. If this is the case, $\mathcal{C}$ returns $T^{d_i^{-1}}$ as the solution for the instance of the *GBDH* problem and stops.
(2) C goes through the list L_2 with entries (U,T,R,ID,PK,Δ,h) for different values of *h*. If such a tuple exists, it returns *h*. Otherwise, it calls the decision bilinear Diffie–Hellman oracle on the tuple $(aP,H_1(ID),U,T)$. If it returns *1* and $e(U,PK) = e(P,R)$, then it returns a random value *h* and updates the list L_2 with a tuple containing the input and *h*.

Signcrypt queries: For each new query (m,Δ,ID_i,ID'), $\mathcal{C}$ proceeds as follows:

- If $ID_i \neq ID_l$, $\mathcal{C}$ signcrypts *m* as follow.
  - If the public key of ID_i has been replaced:
    (1) obtains PK_i and PK' by calling RequestPublicKey oracle on ID_i and ID', respectively,
    (2) chooses a random number $r \in Z_q^*$ and computes $U = rP$, $T = e(Mpk,H_1(ID'))^r$,
    (3) computes $h = H_2(U,T,rPK',ID',PK',\Delta)$,
    (4) computes $V = h \oplus m$,

(5) computes $H = H_3(U,V,ID_i,PK_i,ID', PK')$,
(6) obtains (Δ,s,sP) by calling H_4 oracle on Δ and computes $W = D_i + rH + sPK_i$ (note that $\mathcal{C}$ can obtain D_i from ExtractPartialPrivateKey oracle),
(7) returns $c = (U,V,W)$.

- Otherwise, C signcrypts m in the usual manner by using x_i (obtained from the ExtractSecretValue oracle) and D_i (obtained from ExtractPartialPrivateKey oracle).

- If $ID_i = ID_l$ (and hence $ID' \neq ID_l$), $\mathcal{C}$ performs the following:
(1) obtains PK_i and PK' by calling RequestPublicKey oracle on ID_i and ID', respectively,
(2) generates two random values $u, v \in Z_q^*$ and sets $U = vaP$,
(3) calls H_1 with ID' to obtain (j, ID', r') and computes $T = e(U,r'Mpk)$,
(4) obtains h by calling H_2 on (U,T,R,ID', PK',Δ) and computes $V = m \oplus h$,
(5) defines the hash value $H_3(U,V,ID_l,PK_i, ID',PK')$ as $H = v^{-1}(uP - Q_{ID_l})$, aborting if such a H_3 query has been responded with a different value before. This means that $\mathcal{C}$ updates the list L_3 with $(U,V,ID_l, PK_i,ID',PK',*,H)$.
(6) obtains (Δ,s,sP) by calling H_4 oracle on Δ and sets $W = uaP + sPK_i$ and returns (U,V,W). Note that this is a valid signcryption.

AggregateUnsigncrypt queries: For each new query $\left(U_1,\ldots,U_n,V_1,\ldots,V_n,W,\{ID_i\}_{i=1}^n,ID',\Delta\right)$, $\mathcal{C}$ proceeds as follows:

(1) It queries H_1 and RequestPublicKey oracles with $\{ID_1, \ldots,ID_n,ID'\}$ to obtain $\left\{Q_{ID_1},\ldots,Q_{ID_n}, Q_{ID'}\right\}$ and $\{PK_1, \ldots,PK_n,PK'\}$. $\mathcal{C}$ then executes AggregateVerify algorithm and returns $\perp$ if the verification does not succeed.
(2) For $i = 1,\ldots,n$ (we suppose that A_I replaced the public key of ID'; otherwise, response to this query is more simple):
- If $ID' \neq ID_l, \mathcal{C}$ performs the following:
  (a) calculates $T_i = e(rU_i,Mpk)$, where (j,ID', r) is obtained by calling H_1 on ID',
  (b) goes through L_2 and looks for a tuple $(U_i,T_i, *, ID',PK',\Delta,h)$. If such an entry exists, $\mathcal{C}$ decrypts it using the hash value h. Otherwise, it places the entry $(U_i,T_i,*,ID',PK', \Delta,h)$ for a random value h on the list L_2 and decrypts using this h.
- If $ID' = ID_l$
  (a) then the pairing cannot be calculated. To return a consistent answer, $\mathcal{C}$ goes through L_2 and looks for a tuple $(U_i,*,*,ID',PK',\Delta,h)$. If such an entry exists, $\mathcal{C}$ decrypts it using the hash value h.
  (b) If $\mathcal{C}$ reaches this point of execution, it places the entry $(U_i,*,*,ID',PK',\Delta, h)$ for a random value h on the list L_2 and decrypts using this h. The symbol * denotes an unknown value.

Eventually, A_I outputs two message sets $M_0^* = \left\{M_{0i}^*\right\}_{i=1}^n$ and $M_1^* = \left\{M_{1i}^*\right\}_{i=1}^n$, some state information Δ^* and $n+1$ identities $\left\{ID_i^*\right\}_{i=1}^n$ and ID_R^*. $\mathcal{C}$ places a query on H_1 with input ID_R^*. If the index of ID_R^* is not $l, \mathcal{C}$ fails. Otherwise, it proceeds to construct a challenge as follows. It obtains from L_K the public keys $\left\{PK_i^*\right\}_{i=1}^n$ corresponding to $\left\{ID_i^*\right\}_{i=1}^n$. Then, it sets $\left\{U_i^* = d_icP\right\}_{i=1}^n$ with randomly selected $\left\{d_i \in Z_q^*\right\}_{i=1}^n$ and selects a random bit μ. $\mathcal{C}$ obtains hash values $\{h_i\}_{i=1}^n$ from the H_2 oracle and sets $\left\{V_i^* = M_{\mu i}^* \oplus h_i\right\}_{i=1}^n$. $\mathcal{C}$ then computes $\left\{W_i^* = D_i^* + r_iH_i + x_i^*\, H' = D_i^* + t_iU_i^* + sPK_i^*\right\}_{i=1}^n$, where t_i is obtained from H_3 oracle, s is obtained from H_4 oracle, and D_i^* is calculated by calling the ExtractPartialPrivateKey oracle on ID_i^*. $\mathcal{C}$ now applies the Aggregate algorithm and sends the output (i.e., $c^* = \left(U_1^*,\ldots,U_n^*, V_1^*,\ldots,V_n^*,W^*\right)$) to A_I. Note that since $\left\{ID_i^* \neq ID_R^*\right\}_{i=1}^n$, the ExtractPartialPrivateKey oracle simulation always gives $\mathcal{C}$ the correct value of D_i^*.

Queries made by A_I during Phase 2 are treated as in Phase 1. Finally, A_I will output the index of the message set, which he thinks is signcrypted inside the challenge. Note that from adversary's viewpoint, each index i has the same probability, and thus, the probability that the adversary outputs a particular identity is the same for all identities. If $ID_R^* = ID_l$, the simulation is perfect unless the adversary queries H_2 on one of the challenge-related tuples $\left\{\left(U_i^*,T_i^*,R_i^*,ID_R^*,PK_R^*,\Delta^*\right)\right\}_{i=1}^n$. Given that the hash function H_2 is modeled as a random oracle, the adversary will not have any advantage if one of these tuples does not appear on L_2. However, if this happens, $\mathcal{C}$ solves the *GBDH* problem for the given input (due to the first step in the simulation of H_2) with a probability dependent on the advantage of A_I.

Lemma 2. *The proposed CLASC scheme is secure during Game II, assuming that the CDH problem is hard in* G_1.

Proof. We suppose that a Type *II* adversary A_{II} for our scheme exists. Let $\mathcal{C}$ be a *CDH* attacker who receives a random instance (P,aP,bP) of the *CDH* problem in G_1. We now demonstrate how $\mathcal{C}$ is able to use A_{II} during Game *II* to compute abP. □

$\mathcal{C}$ generates a master key pair (msk,Mpk) and sends G_1, G_2,e,P,Mpk and msk to A_{II}. It chooses an index $l \leqslant q_{ReqPK}$ at random, where q_{ReqPK} is the maximum number of queries that A_{II} could place to RequestPublicKey oracle. $\mathcal{C}$ answers to A_{II} queries as follows.

H_1 *queries:* On the i-th non-repeated query ID, $\mathcal{C}$ chooses $r \in Z_q^*$ uniformly at random and sets $Q_{ID} = rP$. It then adds (ID,r) to the list L_1 and returns Q_{ID}.

RequestPublicKey queries: On the i-th non-repeated query ID, if $i \neq l, \mathcal{C}$ generates a new key pair (x, PK), updates the list L_K with (i,ID,PK,x) and returns PK. If $i = l$, $\mathcal{C}$ returns aP and adds $(l,ID,aP, \perp)$ to L_K. From this point on, we denote the l-th non-repeated identity queried to this oracle with ID_l.

ExtractSecretValue queries: For each new query ID, $\mathcal{C}$ calls RequestPublicKey on ID to obtain (i,ID,PK,x). If $i = l$, $\mathcal{C}$ aborts. Otherwise, it returns x.

H_3 *queries:* On input (U, V, ID, PK, ID', PK'), $\mathcal{C}$ searches L_3 for the entry (U,V,ID,PK,ID',PK',t,tP). If no entry is found, $\mathcal{C}$

generates a random value t in Z_q^*, updates the list L_3 with the input, t and tP. In both cases, $\mathcal{C}$ returns tP.

H_4*queries:* On input (Δ), $\mathcal{C}$ searches L_4 for the entry (Δ,s,sP). If no entry is found, $\mathcal{C}$ generates a random value s in Z_q^*. It then inserts the tuple (Δ,s,sP) in the list L_4. In both cases, $\mathcal{C}$ returns sP.

H_2 *queries:* On input (U,T,R,ID,PK,Δ), $\mathcal{C}$ proceeds as follows:

(1) Checks if $e(aP,d_ibP) = e(P,R)$ for $i = 1,\ldots,n$. If so, $\mathcal{C}$ returns $d_i^{-1}R$ and stops.
(2) Checks the list L_2 for the tuple (U,T,R,ID,PK,Δ,h) for some value of h. If such a tuple exists, $\mathcal{C}$ returns h to A_{II}. Otherwise, if $e(U,mskH_1(ID)) = T$ and $e(U,PK) = e(P,R)$; then, it returns a random value h and updates the list L_2 with a tuple containing the input and h.

Signcrypt queries: For each new query (m,Δ,ID_i,ID'), $\mathcal{C}$ proceeds as follows:

- If $ID_i \neq ID_l$, $\mathcal{C}$ obtains the secret value x_i from the RequestPublicKey and signcrypts the message m in the usual way. $\mathcal{C}$ returns the ciphertext to A_{II}.
- If $ID_i = ID_l$, $\mathcal{C}$ then performs the following:
  (1) obtains PK_l and PK' by calling RequestPublicKey oracle on ID_l and ID', respectively.
  (2) generates a random value $r \in Z_q^*$, sets $U = rP$ and calculates $T = e(U,mskH_1(ID'))$ and $R = rPK'$.
  (3) calls H_2 on (U,T,R,ID',PK',Δ), obtains h and computes $V = m \oplus h$.
  (4) obtains (Δ,s,sP) by calling H_4 on Δ and computes $W = D_l + rH_3(U,V,ID_l,PK_l,ID',PK') + x_lH_4(\Delta) = D_i + rH_3(U,V,ID_l,PK_l,ID',PK') + saP$,
  (5) returns (U,V,W).

AggregateUnsigncrypt queries: For each new query $(U_1,\ldots,U_n,\ V_1,\ldots,V_n,\ W,\{ID_i\}_{i=1}^n,ID',\Delta)$, $\mathcal{C}$ proceeds as follows:

(1) It executes the AggregateVerify algorithm after obtaining $\{Q_{ID_1},\ldots,Q_{ID_n},\ Q_{ID'}\}$ and $\{PK_1,\ldots,PK_n,PK'\}$ with calls to H_1 and RequestPublicKey oracles. It returns $\perp$ if the verification does not succeed.
(2) For $i = 1,\ldots,n$:
  - If $ID' \neq ID_l$, it unsigncrypts in the usual way.
  - If $ID' = ID_l$, then
    (a) It calculates $T_i = e(U_i,r'Mpk)$, where (ID',r') is obtained from L_1.
    (b) $\mathcal{C}$ is unable to compute the correct value of R. To answer the query consistently, $\mathcal{C}$ searches L_2 looking for a tuple $(U_i,T_i,R,ID_l,PK',\Delta,h)$, for different values of R, such that $e(U_i,PK') = e(P,R)$. If such an entry exists, the correct value of R is found, and $\mathcal{C}$ decrypts using h.
    (c) If $\mathcal{C}$ reaches this point of execution, $\mathcal{C}$ places the entry $(U_i,T_i,*,ID',PK',\Delta,h)$ for a random value h on list L_2 and decrypts using this h.

Eventually, A_{II} outputs two message sets $M_0^* = \{M_{0i}^*\}_{i=1}^n$ and $M_1^* = \{M_{1i}^*\}_{i=1}^n$, some state information Δ^* and $n+1$ identities $\{ID_i^*\}_{i=1}^n$ and ID_R^*. $\mathcal{C}$ places a query on H_1 with input ID_R^*. If the index of ID_R^* is not l, $\mathcal{C}$ fails. Otherwise, it proceeds to construct a challenge as follows. It obtains from L_K the public keys $\{PK_i^*\}_{i=1}^n$ corresponding to $\{ID_i^*\}_{i=1}^n$. Then, it sets $U_i^* = d_ibP$ with randomly selected $\left\{d_i \in Z_q^*\right\}_{i=1}^n$ and selects a random bit μ. $\mathcal{C}$ obtains hash values $\{h_i\}_{i=1}^n$ from the H_2 oracle and sets $V_i^* = M_{\mu i}^* \oplus h_i$. The components W_i^* are set to be $D_i^* + r_iH_i + x_i^*H' = D_i^* + t_id_ibP + sPK_i^*$, where t_is are obtained from L_3, s from L_4 and D_i^*s by calling the ExtractPartialPrivateKey oracle on ID_i^*s.

In the second phase, A_{II} performs new queries, which are treated in the same manner as Phase *1*. At the end of the simulation, it will output the index of the message set, which he thinks is signcrypted inside the challenge.

As before, all indices have the same probability from the adversary's viewpoint, and therefore, the probability that the adversary outputs an identity ID_l with index l is the same for all identities. If $ID_R^* = ID_l$, the simulation is perfect unless the adversary queries H_2 on one of the challenge-related tuples $\left\{\left(U_i^*, T_i^*, R_i^*, ID_R^*, PK_R^*, \Delta^*\right)\right\}_{i=1}^n$. Since the hash function H_2 is modeled as a random oracle, the adversary will not have any advantage if one of these tuples does not appear on L_2. However, if this happens, $\mathcal{C}$ solves the *CDH* problem for the given input (due to the first step in the simulation of H_2). Therefore, if A_{II} has any advantage in winning this game, $\mathcal{C}$ can solve *CDH* problem with a probability dependent on the advantage of A_{II}.

Theorem 2. *The proposed certificateless aggregate signcryption scheme is existentially unforgeable against chosen message attack (EUF-CLASC-CMA) in the random oracle model under the CDH intractability assumption and presence of a decision bilinear Diffie–Hellman oracle.*

This theorem follows Lemmas 1 and 2.

Lemma 3. *The proposed CLASC scheme is secure during Game III, assuming that the GDH′ intractability assumption holds.*

Proof. Let $\mathcal{C}$ be a *GDH′* attacker who receives a random instance (P,aP,bP) of the *GDH′* problem in G_1. Let A_I be a type *I* adversary who interacts with $\mathcal{C}$ as modeled in Game *III*. We demonstrate how $\mathcal{C}$ may use A_I to solve the *GDH′* problem, i.e., to compute abP. □

Setup: $\mathcal{C}$ runs the setup algorithm of the proposed scheme and sets $Mpk = aP$. It chooses a random number $l \leqslant q_{H_1}$ as the challenge identity, where q_{H_1} is the maximum number of queries that A_I could place to H_1 oracle. It then sends G_1 ,G_2,e,P and Mpk to A_I. A_I can make some queries (as explained in Game *I*), and $\mathcal{C}$ answers these queries (as in lemma 1), except H_2 queries, which are as follows:

H_2 *queries:* On input (U,T,R,ID,PK,Δ), $\mathcal{C}$ checks if the decisional bilinear Diffie–Hellman oracle returns *1* on $(U,Mpk,H_1(ID),T)$, and checks $e(U,PK) = e(R,P)$. If it is the case, $\mathcal{C}$ checks the list L_2 for the tuple (U,T,R,ID,PK,Δ,h) for some value of h. If such a tuple exists, $\mathcal{C}$ returns h to A_I; otherwise, $\mathcal{C}$ randomly chooses $h \in \{0,1\}^k$ and updates L_2 with (U,T,R,ID,PK,Δ,h). $\mathcal{C}$ also sends h to A_I.

Eventually, A_I returns a set of $n+1$ users with identities $\{ID_1^*,\ldots,ID_n^*,ID_R^*\}$, the corresponding public keys $\{PK_1^*,\ldots,$

$PK_n^*, PK_R^*\}, n$ messages $\{M_i^*\}_{i=1}^n$, some state information Δ^* and a forged aggregate ciphertext $c^* = (U_1^*, \ldots, U_n^*, V_1^*, \ldots, V_n^*, W^*)$. It is required that $ID_l \in \{ID_1^*, \ldots, ID_n^*\}$. Given that each index i has the same probability from the adversary's viewpoint, all identities in the list have the same probability to get picked up. If this event occurs, the simulation is perfect. Without loss of generality, we let $ID_l = ID_1^*$. In addition, the AggregateVerify algorithm should return true on forged aggregate signcryption ciphertext, namely:

$$e(W^*, P) = e\left(\sum_{i=1}^{n} Q_i^*, Mpk\right) \prod_{i=1}^{n} e(H_i, U_i^*) e\left(H', \sum_{i=1}^{n} PK_i^*\right),$$

where $Q_i^* = H_1(ID_i^*), H_i = H_3(U_i^*, V_i^*, ID_i^*, PK_i^*, ID_R^*, PK_R^*)$ and $H' = H_4(\Delta^*)$.

$\mathcal{C}$ recovers (i, ID_i^*, r_i) from $L_1, (U_i^*, V_i^*, ID_i^*, PK_i^*, ID_R^*, PK_R^*, t_i, t_iP)$ from L_3 list, (Δ^*,s,sP) from L_4 list. Now, $\mathcal{C}$ can solve GDH' problem as follows:

$$abP = W^* - \sum_{i=2}^{n} r_i Mpk - \sum_{i=1}^{n} t_i U_i^* - \sum_{i=1}^{n} sPK_i^*.$$

Lemma 4. *The proposed CLASC scheme is secure during Game IV, assuming that the CDH intractability assumption holds.*

Proof. Let $\mathcal{C}$ be a *CDH* attacker who receives a random instance (P,aP,bP) of the *CDH* problem in G_1. Let A_{II} be a type *II* adversary who interacts with $\mathcal{C}$ as modeled in Game *IV*. We demonstrate how $\mathcal{C}$ may use A_{II} to solve the *CDH* problem, i.e., to compute abP. □

Setup: $\mathcal{C}$ generates a master key pair (msk,Mpk) and sends G_1,G_2,e,P,Mpk and msk to A_{II}. It chooses a random number $l \leqslant q_{ReqPK}$ as the challenge identity, where q_{ReqPK} is the maximum number of queries that A_{II} could place to RequestPublicKey oracle. A_{II} can make some queries as explained in Game *II*, and $\mathcal{C}$ answers as in lemma 2, except for the following queries:

H_2 *queries:* On input $(U, T, R, ID, PK, \Delta)$, $\mathcal{C}$ checks if the decisional bilinear Diffie–Hellman oracle returns 1 on (U,Mpk,$H_1(ID)$,T), and checks $e(U,PK) = e(R,P)$. If it is the case, $\mathcal{C}$ checks the list L_2 for the tuple (U,T,R,ID,PK,Δ,h) for some value of h. If such a tuple exists, $\mathcal{C}$ returns h to A_{II}; otherwise, $\mathcal{C}$ randomly chooses $h \in \{0,1\}^k$ and updates L_2 with (U,T,R,ID,PK,Δ,h). $\mathcal{C}$ also sends h to A_{II}.

H_4*queries:* On input (Δ), $\mathcal{C}$ searches L_4 for the entry (Δ,s,sbP). If no entry is found, $\mathcal{C}$ generates a random value s in Z_q^*. It then inserts the tuple (Δ,s,sbP) in the list L_4. In both cases, $\mathcal{C}$ returns sbP.

Signcrypt queries: For each new query (m,Δ,ID_i, ID'), $\mathcal{C}$ calls the RequestPublicKey oracle on ID_i and proceeds as follows:

- If $ID_i \neq ID_l$, $\mathcal{C}$ simply signcrypts the message, getting the secret value x_i from the ExtractSecretValue oracle.
- If $ID_i = ID_l$, $\mathcal{C}$ does the following:
  (1) Generates two random values $u, v \in Z_p^*$, sets $U = vaP$ and calculates $T = e(U, mskQ_{ID'})$.
  (2) Calls H_2 on (U,T,R,ID',PK', Δ) and uses the returned value (h) to compute $V = m \oplus h$, where PK' is obtained by calling the RequestPublicKey oracle on ID'.
  (3) Defines the hash value $H_3(U,V,ID_l,PK_l, ID',PK')$ as $H = v^{-1}(uP - H')$, aborting if such a H_3 query has been responded with a different value before, where $H' = H_4(\Delta)$. This means that $\mathcal{C}$ updates list L_3 with tuple (U,V,ID_l,PK_l,ID',PK', *,H). Finally, $\mathcal{C}$ sets $W = D_i + uaP$ and returns (U,V,W), where $D_i = mskH_1(ID_l)$. Note that this is a valid signcryption.

Eventually, A_{II} returns a set of $n + 1$ users with identities $\{ID_1^*, \ldots, ID_n^*, ID_R^*\}$, the corresponding public keys $\{PK_1^*, \ldots, PK_n^*, PK_R^*\}, n$ messages $\{M_i^*\}_{i=1}^n$, some state information Δ^* and a forged aggregate ciphertext $c^* = (U_1^*, \ldots, U_n^*, V_1^*, \ldots, V_n^*, W^*)$. It is required that $ID_l \in \{ID_1^*, \ldots, ID_n^*\}$. Here too, all indices are the same from adversary's viewpoint, and the probability that $ID_l \in \{ID_1^*, \ldots, ID_n^*\}$ is the same for each l. If this event occurs, the simulation is perfect. Without loss of generality, we let $ID_l = ID_1^*$. In addition, the forged aggregate signcryption ciphertext must be verified by AggregateVerify, namely:

$$e(W^*, P) = e\left(\sum_{i=1}^{n} Q_i^*, Mpk\right) \prod_{i=1}^{n} e(H_i, U_i^*) e\left(H', \sum_{i=1}^{n} PK_i^*\right),$$

Where $Q_i^* = H_1(ID_i^*), H_i = H_3(U_i^*, V_i^*, ID_i^*, PK_i^*, ID_R^*, PK_R^*)$ and $H' = H_4(\Delta^*)$.

$\mathcal{C}$ recovers (ID_i^*, r_i) from L_1, $(U_i^*, V_i^*, ID_i^*, PK_i^*, ID_R^*, PK_R^*, t_i, t_iP)$ from L_3 list, (Δ^*,s,H') from L_4 and $(i, ID_i^*, x_i^*, PK_i^*)$ from L_k and by use of these values can solve *CDH* problem as follows:

$$abP = \frac{W^* - \sum_{i=1}^{n} mskr_iP - \sum_{i=1}^{n} t_iU_i^* - \sum_{i=2}^{n} x_i^* H'}{s}.$$

6.3. Performance analysis

In this section, we analyze the efficiency of the proposed method. To the best of our knowledge, the proposed scheme is the first CLASC scheme proposed in the literature. Therefore, we compare our scheme with Barbosa and Farshim's (BF) scheme (used n times). The comparison is performed in terms of computation complexity and communication load. The results indicate the efficiency of the proposed method. We summarize the results in Table 1 where the following notations are used:

$T_{G_1}^{S-Mul}$: computation time for a scalar point multiplication in an additive group (like G_1),

$T_{G_2}^{Exp}$: computation time for an exponentiation in a multiplicative group (like G_2),

T_P: computation time of one pairing operation,

T_h: computation time of one hash operation,

BF: using Barbosa and Farshim's scheme for each one of the senders separately.

In the proposed scheme, we have aggregated the signature parts of ciphertexts. This makes the communication overhead of the proposed scheme $(n-1)|G_1|$ bits less than using n times Barbosa and Farshim's (BF) scheme. In other words, sending aggregate ciphertext provides a sufficient amount of efficiency over sending each ciphertext separately. Note that it is not possible to reduce the communication overhead of a CLASC scheme to a constant value because two parts of each ciphertext are needed for decryption. Our scheme uses the signcryption

Table 1 Performance analysis.

Scheme	The receiver side computational complexity	Communication complexity
Ours	$(2n+1)T_h + (2n+3)T_P + nT_{G_1}^{S-Mul}$	$(n+1)\|G_1\| + nk$
BF	$(3n)T_h + (5n)T_P + nT_{G_1}^{S-Mul}$	$2n\|G_1\| + nk$

algorithm of Barbosa and Farshim's scheme to signcrypt a message. Therefore, the computational power needed by each sender is the same in both schemes. However, on the receiver side, verification of signatures can be performed in a single step rather than verifying each signature separately. This greatly reduces the costs of the verification process as mentioned in Table 1.

7. Conclusion

Existing literature on aggregate signcryption covers mainly an identity-based setting with the well-known key escrow problem. This paper considers certificateless aggregate signcryption schemes and defines a suitable security model for them. A concrete construction of a certificateless aggregate signcryption scheme is also provided. We prove that this construction is secure under the gap Bilinear Diffie–Hellman and computational Diffie–Hellman intractability assumptions in the random oracle model.

References

Al-riyami, S.S., Paterson, K.G., 2003. Certificateless public key cryptography. In: Proceedings of the Asiacrypt'03. LNCS, vol. 2894. Springer-Verlag, pp. 452–473.

Baek, J., Safavi-naini, R., Susilo, W., 2005. Certificateless public key encryption without pairing. In: Computers and Operations Research. Springer-Verlag, pp. 134–148.

Barbosa, M., Farshim, P., 2008. Certificateless signcryption. In: Abe, M., Gligor, V. (Eds.), Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS-08). ACM, New York, pp. 369–372.

Boneh, D., Gentry, C., Lynn, B., Shacham, H., 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (Ed.), EUROCRYPT 2003, LNCS, vol. 2656. Springer-Verlag, Warsaw, Poland, pp. 416–432.

Boyen, X., 2003. Multipurpose identity based signcryption: a swiss army knife for identity based cryptography. In: Boneh, D. (Ed.), Advances in Cryptology-Crypto 2003, LNCS, vol. 2729. Springer-Verlag, Berlin, pp. 383–399.

Chang, S., Wong, D., Mu, Y., Zhang, Z., 2009. Certificateless Threshold Ring Signature. Inf. Sci. 179, 3685–3696.

Chen, L., Malone-Lee, J., 2005. Improved identity-based signcryption. In: Vaudenay, S. (Ed.), Proceedings of the Eighth International Workshop on Theory and Practice in Public Key Cryptography 2005, LNCS, vol. 3386. Springer-Verlag, Berlin, pp. 362–379.

Cheng, X., Liu, J., Guo, L., Wang, X., 2006. Identity-based multisignature and aggregate signature schemes from m-torsion groups. J. Electron. 23, 569–573.

Duan, S., 2008. Certificateless undeniable signature scheme. Inf. Sci. 178, 742–755.

Gentry, C., Ramzan, Z., 2006. Identity-based aggregate signatures. In: Yung, M. et al. (Eds.), PKC 2006, LNCS, vol. 3958. Springer-Verlag, New York, USA, pp. 257–273.

Guoyan, Z., Xiaoyun, W., 2009. Certificateless encryption scheme secure in standard model. Tsinghua Sci. Technol. 14, 452–459.

Hu, B., Wong, D., Zhang, Z., Deng, X., 2007. Certificateless signature: a new security model and an improved generic construction. Design. Code. Cryptogr. 42, 109–126.

Islam, S.H., Biswas, G., 2013a. Certificateless short sequential and broadcast multisignature schemes using elliptic curve bilinear pairings. J. King Saud Univ. Comput. Inf. Sci. 26, 89–97.

Islam, S.H., Biswas, G., 2013b. Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings. J King Saud Univ. Comput. Inf. Sci. 25, 51–61.

Libert, B., Quisquater, J., 2003. New identity based signcryption schemes from pairings. In: Proceedings of the 2003 IEEE Information Theory Workshop, Paris, France. pp. 155–158.

Liu, Z., Hu, Y., Zhang, X., Ma, H., 2010. Certificateless signcryption scheme in the standard model. Inf. Sci. 180, 452–464.

Long, Y., Chen, K., 2007. Certificateless threshold cryptosystem secure against chosen-ciphertext attack. Inf. Sci. 177, 5620–5637.

Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H., 2004. Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J. (Eds.), Eurocrypt 2004, LNCS 3027. Springer-Verlag, Interlaken, Switzerland, pp. 74–90.

Malone-Lee, J., 2002. Identity based signcryption. In: Cryptology ePrint Archive, Report 2002/098 <http://eprint.iacr.org/2002/098>.

Miao, S., Zhang, F., Li, S., Mu, Y., 2013. On security of a certificateless signcryption scheme. Inf. Sci. 232, 475–481.

Selvi, S., Vivek, S., Ragan, C., 2009. On the Security of Certificateless Signcryption Schemes. In: Cryptology ePrint Archive, Report 2009/298 <http://eprint.iacr.org/2009/298>.

Selvi, S., Vivek, S., Shriram, J., Kalaivani, S., Rangan, C., 2009. Identity based aggregate signcryption schemes. In: Lecture Notes in Computer Science 5922 LNCS. pp. 378–397.

Seo, S.-H., Choi, K.Y., Hwang, J.Y., Kim, S., 2012. Efficient certificateless proxy signature scheme with provable security. Inf. Sci. 188, 322–337.

Shamir, A., 1984. Identity based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (Eds.), Crypto-84, LNCS 196. Springer-Verlag, Santa Barbara, California, USA, pp. 47–53.

Sun, Y., Zhang, F., 2010. Secure certificateless encryption with short ciphertext. Chin. J. Electron. 19, 313–318.

Tso, R., Huang, X., Susilo, W., 2012. Strongly secure certificateless short signatures. J. Syst. Softw. 85, 1409–1417.

Weng, J., Yao, G., Deng, R.H., Chen, M., Li, X., 2011. Cryptanalysis of a certificateless signcryption scheme in the standard model. Inf. Sci. 181, 661–667.

Xiong, H., Guan, Z., Chen, Z., Li, F., 2013. An efficient certificateless aggregate signature with constant pairing computations. Inf. Sci. 219, 225–235.

Xu, J., Zhang, Z., Feng, D., 2005. Id-based aggregate signatures from bilinear pairings. In: Desmedt, Y.G. (Ed.), CANS 2005, LNCS 3810. Springer-Verlag, Shenzhen, China, pp. 110–119.

Zhang, J., Mao, J., 2012. An efficient RSA-based certificateless signature scheme. J. Syst. Softw. 85, 638–642.

Zhang, L., Qin, B., Wu, Q., Zhang, F., 2010. Efficient many-to-one authentication with certificateless aggregate signatures. Comput. Netw. 54, 2482–2491.

Zhang, L., Zhang, F., 2008. Certificateless signature and blind signature. J. Electron. 25, 629–635.

Zhang, L., Zhang, F., 2009. A new certificateless aggregate signature scheme. Comput. Commun. 32, 1079–1085.

Zhao, W., Ye, D., 2012. Certificateless undeniable signatures from bilinear maps. Inform. Sciences 199, 204–215.

Zheng, Y., 1997. Digital signcryption or how to achieve cost (signature and encryption) ⩽ cost (signature) + cost(encryption). In: Goos, G., Hartmanis, J., van Leeuwen, J. (Eds.), . In: Advances in Cryptology-Crypto 1997, LNCS, vol. 1294. Springer-Ver.