IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Provably secure certificateless aggregate signature scheme with designated verifier in an improved security model

ISSN 1751-8709
Received on 10th December 2016
Revised 17th May 2018
Accepted on 19th July 2018
E-First on 19th October 2018
doi: 10.1049/iet-ifs.2018.5226
www.ietdl.org

Limin Shen[1,2,3] ✉, *Jianfeng Ma*[2,3], *Yinbin Miao*[3], *Hai Liu*[3]

[1]*School of Computer Science and Technology, Nanjing Normal University, Nanjing, People's Republic of China*
[2]*School of Computer Science and Technology, Xidian University, Xi'an, People's Republic of China*
[3]*Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an, People's Republic of China*
✉ *E-mail: shenlimin@njnu.edu.cn*

Abstract: An aggregate signature (AS) scheme combines multiple signatures which is generated by many different users into a single one. This feature is very beneficial for diminishing storage cost, bandwidth and verification cost. Many previous attempts have been made for designing AS schemes, while the former security models have not clearly addressed coalition attacks, and most of the existing AS schemes cannot resist these kinds of attacks. In this study, the authors provide a modified security model of certificateless AS (CLAS) schemes and then give a new CLAS scheme. The security of their present scheme can be rigorously proved based on the computational Diffie–Hellman assumption in the random oracle model. Furthermore, their scheme can resist such coalition attacks, i.e. an AS in their scheme is valid iff all single signatures used to generate the AS are valid.

1 Introduction

Certificateless public key cryptography (CL-PKC) [1] is an absorbing research field because it is free from the key escrow problem. In Asiacrypt 2003, Al-Riyami and Paterson [1] first introduced the CL signature (CLS) scheme. Later, its security weakness was shown by Huang *et al.* [2]. After that, many practical and efficient schemes about CL-PKC have been proposed [3–7].

In Eurocrypt 2003, Boneh *et al.* [8] presented the conception of *aggregate signatures (ASs)*, an AS scheme can combine many signatures generated by many different users into a single one, thus the communication cost for transmitting these signatures can be decreased. So, it is very valid in compressing signatures, diminishing bandwidth and storage cost, and can be a decisive building block in some situations. Hence, designing secure and efficient AS schemes has been an interesting research field. Aggregated signatures are being applied to many application environments such as data collecting in sensor networks [9, 10], secure communication between vehicles and the infrastructure in vehicular *ad hoc* networks [11], distributed aggregate privacy-preserving authentication [12] and so on. So far, there are a lot of AS schemes which have been available in various settings such as traditional public key setting (public key infrastructures) [13–16], identity-based PKC [17–19] and CL-PKC [20–24]. Unluckily, many of the existing AS schemes cannot resist one kind of practical and powerful attacks called coalition attacks [24].

By coalition attack we mean some signers collude to generate a valid AS by using some individual signatures contained invalid ones. Till now, only very few aggregate schemes [9, 24–26] have touched on the coalition attacks, while the former security models of AS schemes have not clearly described these kinds of attacks. We note that once such an attack is successful, a valid AS may fail to provide a guarantee for the validity of all individual signatures. This fact obviously breaches the primary security goal for AS schemes, because secure AS schemes should require that an AS is valid iff each individual signature is valid. Actually, in the former security models, the adversary can get control of some signers, so we think coalition attacks are really practical. Therefore, such coalition attacks should be taken into account in an appropriate security model.

In most of the former security models, the adversaries merely forge a single signature of a signer. This implies that most of the attacks just focus on the involved basic signature scheme. A secure AS scheme not only requires the basic signature scheme should be existentially unforgeable, but also the aggregate algorithm should withstand various possible attacks. In this paper, we provide a new CLAS scheme (Fig. 1) which can be proved secure in accordance with an improved security model. The security model's highlight is that the only way to generate a valid CLAS is that all individual ones involved are valid. Thus, the adversary is endowed with sufficient capacity to launch any coalition attacks. We say an adversary succeeds if the adversary can generate a valid CLAS using some individual signatures including invalid ones.

The following section briefly introduces the preliminaries for CLAS schemes. Section 3 provides an improved security model for CLAS schemes. Our secure CLAS scheme with designated verifiers is given in Section 4. Then, in Section 5, we present a detailed security proof of our scheme in the random oracle model based on the computational Diffie–Hellman (CDH) assumption. The performance analysis of our CLAS scheme is given in Section 6. At last, Section 7 gives the conclusion.

2 Preliminaries

This section revisits some basic notions, containing the definition of bilinear pairing, the necessary complexity assumptions and the outline of CLAS.

2.1 Bilinear pairing and complexity assumptions

Let $\mathbb{G}$ be a cyclic group with a generator P and let $\mathbb{G}_T$ denote a cyclic group of the same prime order p with $\mathbb{G}$.

Definition 1: If a map $\hat{e}: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfies the following three properties, then it is called a bilinear pairing:

1. *Bilinear*: for all $P_1, P_2 \in \mathbb{G}$ and $\tau, \upsilon \in Z_p^*$, $\hat{e}(\tau P_1, \upsilon P_2) = \hat{e}(P_1, P_2)^{\tau\upsilon}$.
2. *Non-degenerate*: $\hat{e}(P, P) \neq 1_T$, where 1_T is $\mathbb{G}_T$'s identity element.

3. *Computable*: for all $P, Q \in \mathbb{G}$, $\hat{e}(P, Q)$ is efficiently computable.

Definition 2: CDH problem: Given $P, \tau P, \nu P \in \mathbb{G}$, to compute $\tau\nu P \in \mathbb{G}$.

If there is no algorithm that solves the CDH problem running in time t with advantage at least ϵ, we say that the (t, ϵ)-CDH assumption holds in $\mathbb{G}$.

2.2 Certificateless AS

Normally, a CLAS scheme comprises an *Aggregate* algorithm, an *AggregateVerify* algorithm and a basic CLS scheme. In general, a CLAS scheme contains nine probabilistic polynomial time (PPT) algorithms: *Setup, PartialPrivateKeyExtract, SetSecretValue, SetPrivateKey, SetPublicKey, Sign, Verify, Aggregate and AggregateVerify*.

Setup: Key generation centre (KGC) generates the master secret key msk and the system parameters *param* when it is provided a security integer parameter l.

PartialPrivateKeyExtract: KGC generates the ID's partial private key S_{ID} when it is given an identity ID.

SetSecretValue: A user ID generates its secret value r_{ID}.

SetPrivateKey: A user ID generates its private key $\mathrm{SK}_{\mathrm{ID}}$.

SetPublicKey: A user ID generates its public key $\mathrm{PK}_{\mathrm{ID}}$.

Sign: A signer with the identity ID, private key $\mathrm{SK}_{\mathrm{ID}}$ and a message M generates a signature σ_{ID}.

Verify: A verifier verifies a signature σ_{ID} on $(M, \mathrm{ID}, \mathrm{PK}_{\mathrm{ID}})$, it outputs *accept* if σ_{ID} is valid or $\perp$ otherwise.

Aggregate: Let an aggregating set $\mathbb{U} = \{\mathfrak{U}_1, \ldots, \mathfrak{U}_n\}$, and $\mathfrak{U}_i$'s identity is ID_j with the corresponding public key $\mathrm{PK}_{\mathrm{ID}_j}$, $j = 1, \ldots, n$. An aggregator outputs the AS σ when it is given the message–signature pairs (M_j, σ_j), $j = 1, \ldots, n$.

AggregateVerify: A verifier verifies an AS σ on $\{(M_j, \mathrm{ID}_j, \mathrm{PK}_{\mathrm{ID}_j}),\ j = 1, \ldots, n\}$, it outputs *accept* if σ is valid or $\perp$ otherwise.

3 Improved security model with a designated verifier

The AS's motivation is to compress many single signatures generated by individual users into an AS. This indicates that the only way to generate a valid AS is to input valid single signatures into the aggregate algorithm. Obviously, the basic security requirement is existentially unforgeable for an AS scheme. That is to say, the *Aggregate* algorithm and the involved basic signature scheme are required to be existentially unforgeable. Therefore, a secure CLAS scheme should meet the following two properties.

3.1 Basic CLS scheme's security

A CLS scheme should hold secure even if the adversary has gotten the target identity's secret value or partial private key. So, there are two types of adversaries [3]: Type I adversaries marked by $\mathfrak{A}_{\mathrm{I}}$ having the ability to replace any identity's public key with new values of their choice, cannot access to the master secret key; Type II adversaries marked by $\mathfrak{A}_{\mathrm{II}}$ can access to the system master secret key, cannot perform the public key replacement; here, $\mathfrak{A}_{\mathrm{II}}$ imitates an *honest-but-curious* KGC. For both types of adversaries, the involved basic CLS scheme should be secure.

Definition 3: existentially unforgeable against adaptively chosen message attacks *(EUF-CMA) secure*: A CLS scheme is said to be EUF-CMA secure if there is no PPT adversary, $\mathfrak{A}_{\mathrm{I}}$ or $\mathfrak{A}_{\mathrm{II}}$, who can break the scheme with a non-negligible advantage.

For space limitation, we omit the detailed security model for CLS schemes.

3.2 Security of the aggregate algorithm

The aggregate algorithm should withstand coalition attacks. By now, many CLAS schemes [21–24] have been presented, unfortunately, most of them only consider the existential unforgeability of the involved basic signature scheme. Very few papers [24, 26] have touched on the security of aggregate algorithm. So, we emphatically consider the security of aggregate algorithm in this paper following the framework [24]. Adversaries in the former security model cannot access all the private keys, so their capacities are limited. We improve the security model mentioned in [24], the adversary can access the relevant oracles to gain every signer' private key, so we do not distinguish $\mathfrak{A}_{\mathrm{I}}$ and $\mathfrak{A}_{\mathrm{II}}$ adversary.

Another feature of the improved security model is that only the designated verifier can verify the AS's validity. Then, aggregate verification oracle is needed in the improved security model, which

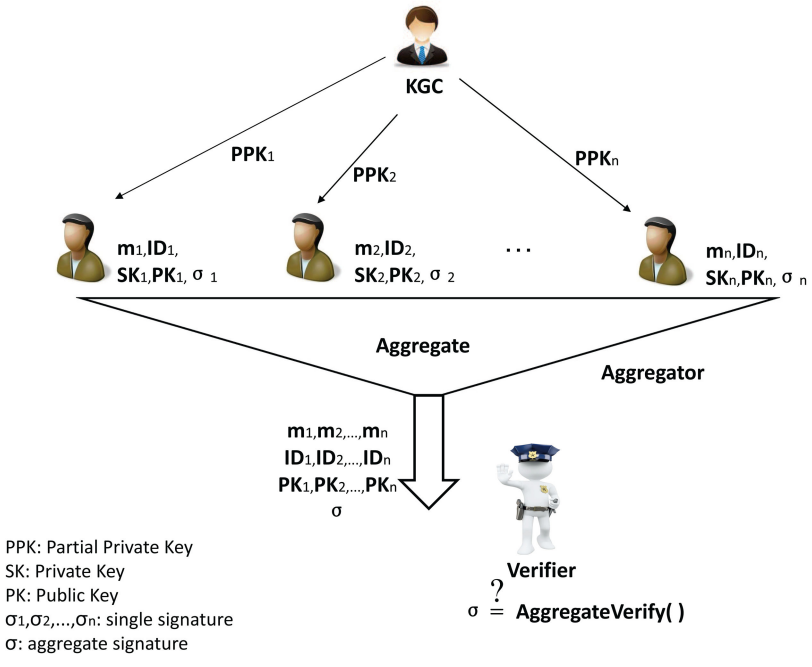

Fig. 1 *CLAS scheme*

is controlled by the challenger to answer some queues from the adversary.

The following is the game between an adversary $\mathfrak{A}$ and a challenger $\mathfrak{C}$. The game consists of three steps: Stepup, Queries and Forgery:

1. *Setup*: The challenger $\mathfrak{C}$ generates the system parameters *param* and the master secret key msk when inputting the security parameter l. Moreover, $\mathfrak{C}$ randomly generates the public–private key pair (PK_{ver}, SK_{ver}) for a designated verifier, then $\mathfrak{C}$ gives $\mathfrak{A}$ PK_{ver} and *param*.
2. *Queries*: A general adversary $\mathfrak{A}$ can make the following queries:
   - *Private key request query* $\mathcal{O}_{SK}(ID_i)$: $\mathfrak{A}$ requests such a query, $\mathfrak{C}$ generates the particular user's SK_{ID} by running algorithm SetPrivateKey and returns SK_{ID} to $\mathfrak{A}$ (first running PartialPrivateKeyExtract or SetSecretValue if necessary).
   - *Aggregate verification request query* $\mathcal{O}_{AggV}(\{ID_i, m_i, PK_i, i = 1, \ldots, n\}, \sigma)$: $\mathfrak{A}$ requests such a query, $\mathfrak{C}$ answers whether the AS σ is valid for the submitting tuples by running algorithm AggregateVerify.
3. *Forgery*: Finally, $\mathfrak{A}$ outputs its forgery $\left(\{ID_j, m_j, PK_j, \sigma_j, j = 1, \ldots, n\}, \sigma^*\right)$. If the following conditions are satisfied, $\mathfrak{A}$ will win the game:
   - The AS σ^* is valid.
   - At least one single signature σ_j is invalid ($j = 1, \ldots, n$).

That is to say, $\mathfrak{A}$ wins the game iff it forges a valid AS using a group of single signatures which are not all valid. $\mathfrak{A}$'s advantage $Adv_{Agg\ Sig_{\mathfrak{A}}}$ in attacking a CLAS scheme is defined as the probability of success in winning the game.

Definition 4: An adversary $\mathfrak{A}$ is said to (t, n, ϵ) – break a CLAS scheme in the above model if $\mathfrak{A}$ runs in time at most t, forges an AS generated by at most n individual signatures and $Adv_{Agg\ Sig_{\mathfrak{A}}}$ is at least ϵ.

A CLAS scheme is (t, n, ϵ) – secure if no adversary (t, n, ϵ) – breaks it.

4 New CLAS scheme

Let $\mathbb{G}$, $\mathbb{G}_T$, p and $\hat{e}$ be the same ones as defined in Section 2.1. We now describe our proposed CLAS scheme in detail as follows:

Setup: The KGC selects collision resistant hash functions $H_1, H_2, H_3: \{0, 1\}^* \to \mathbb{G}$, and $H: \{0, 1\}^* \to Z_q^*$. It chooses $\alpha \in_R Z_q^*$ and $P \in \mathbb{G}$, sets $msk = \alpha$, $P_0 = \alpha P$ and $param = \{\mathbb{G}, \mathbb{G}_T, \hat{e}, H, H_1, H_2, H_3, q, P, P_0\}$. Then, the designated verifier chooses $s \in Z_q^*$ at random, computes and publishes $PK_{ver} = sP$, its private key is $SK_{ver} = s$.

PartialPrivateKeyExtract: Given ID, the KGC generates $Q_{ID} = H_1(ID)$, $S_{ID} = \alpha Q_{ID}$. It thus gives the partial private key S_{ID} to ID through a secret channel.

SetSecretValue: The user ID generates its secret value r_{ID} by randomly selecting $r_{ID} \in Z_q^*$.

SetPrivateKey: The user ID sets its full private key as $SK_{ID} = (S_{ID}, r_{ID})$.

SetPublicKey: The user ID with secret value r_{ID} sets its public key as $PK_{ID} = r_{ID}P$.

Sign: A user ID with private key SK_{ID} wants to sign a message M, it generates $t \in Z_q^*$ randomly, computes

$$U = tP,$$

$$V = S_{ID} + r_{ID}H_2(U, ID, M, PK_{ID}) + tH_3(U, ID, M, PK_{ID}),$$

then exports the signature $\sigma = (U, V)$.

Verify: When receiving $(M, \sigma, PK_{ID}, ID, param)$, the verifier sets

$$h_2 = H_2(U, ID, M, PK_{ID}),$$

$$h_3 = H_3(U, ID, M, PK_{ID}),$$

$$Q_{ID} = H_1(ID),$$

and then checks the formula

$$\hat{e}(V, P) \stackrel{?}{=} \hat{e}(Q_{ID}, P_0)\hat{e}(h_2, PK_{ID})\hat{e}(h_3, U).$$

The verifier returns *accept* if it holds; otherwise, returns $\perp$.

Aggregate: Let $\mathbb{U} = \{\mathfrak{U}_1, \ldots, \mathfrak{U}_n\}$ be a group of n signers. The identity of $\mathfrak{U}_i$ is ID_j, $j = 1, \ldots, n$. The corresponding public keys are $\{PK_1, \ldots, PK_n\}$, where $PK_j = PK_{ID_j}$. Suppose the message–signature pairs $\{M_j, \sigma_j = (U_j, V_j)\}$ come from $\mathfrak{U}_i$, $j = 1, \ldots, n$. The aggregator computes

$$V' = \sum_{j=1}^{n} V_j,$$

$$h = H(\hat{e}(V_1, PK_{ver}), \ldots, \hat{e}(V_n, PK_{ver})),$$

$$V = hV'.$$

Then it outputs $\sigma = (V, U_1, U_2, \ldots, U_n)$ as the AS.

AggregateVerify: For each $j \in \{1, \ldots, n\}$, the designated verifier with a secret $s \in Z_q^*$ satisfying $PK_{server} = sP$ computes

$$Q_{ID_j} = H_1(ID_j),$$

$$h_{2j} = H_2(U_j, ID_j, M_j, PK_j),$$

$$h_{3j} = H_3(U_j, ID_j, M_j, PK_j),$$

and

$$h' = H(\hat{e}(Q_{ID_1}, sP_0)\hat{e}(h_{21}, sPK_1)\hat{e}(h_{31}, sU_1), \ldots, \hat{e}(Q_{ID_n}, sP_0)\hat{e}(h_{2n}, sPK_n)\hat{e}(h_{3n}, sU_n)),$$

then checks

$$\hat{e}(V, P) \stackrel{?}{=} \prod_{j=1}^{n} \hat{e}(Q_{ID_j}, h'P_0)\hat{e}(h_{2j}, h'PK_j)\hat{e}(h_{3j}, h'U_j).$$

The designated verifier accepts σ if the equation holds; otherwise, rejects it.

Correctness: If

$$\hat{e}(V_j, P) = \hat{e}(Q_{ID_j}, P_0)\hat{e}(h_{2j}, PK_j)\hat{e}(h_{3j}, U_j), \quad j \in \{1, \ldots, n\},$$

then we have

$$\hat{e}(V_j, PK_{ver}) = \hat{e}(Q_{ID_j}, sP_0)\hat{e}(h_{2j}, sPK_j)\hat{e}(h_{3j}, sU_j), \quad j \in \{1, \ldots, n\},$$

and

$$
\begin{aligned}
h' &= H\big(\hat{e}\big(Q_{ID_1}, sP_0\big)\hat{e}(h_{21}, s\mathrm{PK}_1)\hat{e}(h_{31}, sU_1),\\
&\quad \ldots,\\
&\quad \hat{e}\big(Q_{\mathrm{ID}_n}, sP_0\big)\hat{e}(h_{2n}, s\mathrm{PK}_n)\hat{e}(h_{3n}, sU_n)\big)\\
&= H(\hat{e}(V_1, \mathrm{PK}_{\mathrm{ver}}), \ldots, \hat{e}(V_n, \mathrm{PK}_{\mathrm{ver}}))\\
&= h\,.
\end{aligned}
$$

Then

$$
\begin{aligned}
&\hat{e}(V, P)\\
&= \hat{e}\left(h\sum_{j=1}^{n} V_j, P\right)\\
&= \prod_{j=1}^{n} \hat{e}\big(Q_{\mathrm{ID}_j}, hP_0\big)\hat{e}\big(h_{2j}, \mathrm{PK}_j\big)\hat{e}\big(h_{3j}, hU_j\big)\\
&= \prod_{j=1}^{n} \hat{e}\big(Q_{\mathrm{ID}_j}, h'P_0\big)\hat{e}\big(h_{2j}, h'\mathrm{PK}_j\big)\hat{e}\big(h_{3j}, h'U_j\big)\,.
\end{aligned}
$$

5 Security analysis

5.1 Basic signature scheme's security

Theorem 1: Suppose an EUF-CMA adversary $\mathfrak{A}_{\mathrm{I}}$ can break our signature scheme with advantage ϵ, and suppose $\mathfrak{A}_{\mathrm{I}}$ can run in time t, make at most q_{par} times partial private key queries, q_{pk} times public key queries, q_s times sign queries and r_{H_i} times H_i $(1 \le i \le 3)$ random oracle queries. Then, there exists a challenger $\mathfrak{C}$ to solve the CDH problem with probability $\epsilon' \ge \big(1/r_{H_1}\big)\epsilon$ and in time $t' \le t + \big(r_{H_1} + r_{H_2} + r_{H_3} + q_{\mathrm{par}} + q_{\mathrm{pk}} + 5q_s\big)t_{\mathrm{mul}}$, where t_{mul} marks the time needed for a scalar multiplication in $\mathbb{G}$.

Proof: Suppose $\mathfrak{A}_I$ can break our scheme's EUF-CMA security, then the challenger $\mathfrak{C}$ with inputting a random instance (P, aP, bP) can solve the CDH problem. Then, we will show that how $\mathfrak{C}$ can use $\mathfrak{A}_I$ to compute abP in $\mathbb{G}$.

At first, $\mathfrak{C}$ generates $\mathrm{param} = (q, \mathbb{G}, \mathbb{G}_T, P, P_0 = bP, \hat{e}, H_1, H_2, H_3)$. Here, H_1, H_2, H_3 hash functions are considered as random oracles commanded by $\mathfrak{C}$. Moreover, $\mathfrak{C}$ chooses a random integer index $\tau \in [1, r_{H_1}]$, assume H_1's τth query is on ID^*.

$\mathfrak{C}$ responds to H_1, H_2, H_3 queries, partial private key extraction queries, secret value queries, public key queries, public key replacement and sign queries as follows:

H_1 queries: $\mathfrak{C}$ maintains a list H_1^{list}: $(\mathrm{ID}_i, h_{1i}, Q_i)$. $\mathfrak{A}_I$ requests an H_1 query on ID_i, $\mathfrak{C}$ follows the steps below:

- if $i = \tau$, set $h_{1\tau} = \perp$, $Q_\tau = aP$;
- else, $\mathfrak{C}$ chooses $h_{1i} \in Z_q^*$ at random, computes

$$Q_i = H_1(\mathrm{ID}_i) = h_{1i}P;$$

- Add $(\mathrm{ID}_i, h_{1i}, Q_i)$ to the list.

H_2 queries: $\mathfrak{C}$ maintains a list H_2^{list}: $(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}, h_{2i}, h_{2i}P)$. $\mathfrak{A}_I$ requests an H_2 query on $(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i})$, $\mathfrak{C}$ chooses $h_{2i} \in Z_q^*$ at random computes $H_2\big(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}\big) = h_{2i}P$ and adds them to the list.

H_3 queries: $\mathfrak{C}$ maintains a list H_3^{list}: $\big(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}, h_{3i}, h_{3i}P\big)$. $\mathfrak{A}_I$ requests an H_3 query on $\big(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}\big)$, $\mathfrak{C}$ chooses $h_{3i} \in Z_q^*$ at random, computes $H_3\big(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}\big) = h_{3i}P$, and adds them to the list.

Then, we suppose $\mathfrak{A}_I$ invariably makes the suitable H_1 queries before others.

Partial Private Key Extraction queries: $\mathfrak{A}_I$ requests such a query on ID_i, then

- if $i = \tau$, $\mathfrak{C}$ aborts the game and
- else, $\mathfrak{C}$ first submits ID_i to H_1 queries, then searches H_1^{list} for a tuple $(\mathrm{ID}_i, h_{1i}, Q_i)$ and calculates

$$S_{\mathrm{ID}_i} = b\,H_1(\mathrm{ID}_i) = h_{1i}\,bP = h_{1i}\,P_0\,.$$

Secret value queries: $\mathfrak{A}_I$ can query any identity's secret value and $\mathfrak{C}$ just generates $r \in_{\mathrm{R}} Z_q^*$ as the respondence.

Public key queries: $\mathfrak{A}_I$ requests such a query, $\mathfrak{C}$ generates $r \in_{\mathrm{R}} Z_q^*$ and computes $\mathrm{PK}_{\mathrm{ID}} = r\,P$ as the respondence.

Public key replacement: $\mathfrak{A}_I$ can generate a new value to replace any identity's public key.

Sign queries: $\mathfrak{A}_I$ requests such a query on (M, ID_i):

- if ID_i's public key has not been replaced and $i \ne \tau$, $\mathfrak{C}$ produces a signature by normally running the *Sign* algorithm.
- if ID_i's public key has been replaced or $i = \tau$, $\mathfrak{C}$ generates a signature as follows:
  - Chose $u, v \in Z_q^*$ randomly.
  - Compute $U = uP_0$, $V = vP_0 + h_{2i}\mathrm{PK}_{\mathrm{ID}_i}$.
  - $\sigma = (U, V)$ is the signature. Here, we set

$$H_3\big(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}\big) = u^{-1}(vP - H_1(\mathrm{ID}_i))\,.$$

  If there has been a tuple $(U, \mathrm{ID}_i, M, \mathrm{PK}_{\mathrm{ID}_i}, \perp)$, we will repeat this signature procedure by picking another $u \in Z_q^*$.

Forge: Finally, $\mathfrak{A}_I$ outputs a forgery $(\sigma^* = (U^*, V^*), \mathrm{ID}^*, M^*)$. σ^* should satisfy the following verification if it is valid:

$$\hat{e}(V^*, P) = \hat{e}(Q_{\mathrm{ID}^*}, P_0)\hat{e}(h'_2, \mathrm{PK}_{\mathrm{ID}^*})\hat{e}(h'_3, U^*),$$

where

$$h'_2 = H_2(U^*, \mathrm{ID}^*, M^*, \mathrm{PK}_{\mathrm{ID}^*}),$$

$$h'_3 = H_3(U^*, \mathrm{ID}^*, M^*, \mathrm{PK}_{\mathrm{ID}^*})\,.$$

Search the H_2^{list} and H_3^{list} for

$$H_2(U^*, \mathrm{ID}^*, M^*, \mathrm{PK}_{\mathrm{ID}^*}) = h_{2\tau}P,$$

$$H_3(U^*, \mathrm{ID}^*, M^*, \mathrm{PK}_{\mathrm{ID}^*}) = h_{3\tau}P,$$

to get $h_{2\tau}$ and $h_{3\tau}$. Obviously, the verification equation can be converted to

$$\hat{e}(V^* - h_{2\tau}\,\mathrm{PK}_{\mathrm{ID}^*} - h_{3\tau}\,U^*, P) = \hat{e}(abP, P)\,.$$

Now, $\mathfrak{C}$ can easily solve the CDH problem by computing

$$abP = V^* - h_{2\tau}\,\mathrm{PK}_{\mathrm{ID}^*} - h_{3\tau}\,U^*\,.$$

Analysis: We are not hard to get $\mathfrak{C}$'s advantage $\epsilon' \ge \big(1/r_{H_1}\big)\epsilon$ in solving the CDH problem within the time

$$t' \le t + (r_{H_1} + r_{H_2} + r_{H_3} + q_{\mathrm{par}} + q_{\mathrm{pk}} + 5q_s)\,t_{\mathrm{mul}}\,.$$

□

Theorem 2: If there exists an EUF-CMA adversary $\mathfrak{A}_{\mathrm{II}}$ can break the CLS scheme with advantage ϵ, and suppose $\mathfrak{A}_{\mathrm{II}}$ can run in time t, make at most q_{pk} times public key queries, q_s times sign queries and r_{H_i} times H_i $(1 \le i \le 3)$ random oracle queries. Then,

there exists a challenger $\mathfrak{C}$ who can solve the CDH problem with advantage $\epsilon' \geq (1/r_{H_2})\epsilon$ and in time $t' \leq t + (r_{H_1} + r_{H_2} + r_{H_3} + q_{\text{pk}} + 3q_s)\, t_{\text{mul}}$, where t_{mul} marks the time needed for a scalar multiplication in $\mathbb{G}$.

Proof: Suppose $\mathfrak{A}_{\text{II}}$ can break our scheme's EUF-CMA security, then the challenger $\mathfrak{C}$ with inputting a random instance (P, aP, bP) can solve the CDH problem. Then, we will show that how $\mathfrak{C}$ can use $\mathfrak{A}_{\text{II}}$ to compute abP in $\mathbb{G}$.

At first, $\mathfrak{C}$ picks $\alpha \in_{\text{R}} Z_q^*$ provides $\mathfrak{A}_{II}$ with $\text{msk} = \alpha$ and $\text{param} = (p, \mathbb{G}, \mathbb{G}_T, P, \hat{e}, P_0 = \alpha P, H_1, H_2, H_3)$ described as in Section 5. Here, H_1, H_2, H_3 hash functions are considered as random oracles controlled by $\mathfrak{C}$. Moreover, $\mathfrak{C}$ chooses a random integer index $\tau \in [1, r_{H_2}]$, assume H_2's τth query is on ID^*.

$\mathfrak{A}_{\text{II}}$ can make secret values, public keys and sign queries. Moreover, $\mathfrak{C}$ responds these queries. Note that $\mathfrak{A}_{\text{II}}$ can get all partial private keys because it knows msk. Assume $\mathfrak{A}_{\text{II}}$ invariably makes the suitable H_2 queries before others.

H_1 queries: $\mathfrak{C}$ maintains a list H_1^{list}: (ID_i, h_{1i}, Q_i). When receiving an H_1 query on ID_i, $\mathfrak{C}$ chooses $h_{1i} \in_{\text{R}} Z_q^*$, computes $Q_i = H_1(\text{ID}_i) = h_{1i}P$, and adds them to the list.

H_2 queries: $\mathfrak{C}$ maintains a list H_2^{list}: (U, ID_i, M, PK_{ID_i}, h_{2i}, $h_{2i}P$). When receiving an H_2 query on (U, ID_i, M, PK_{ID_i}), $\mathfrak{C}$ chooses $h_{2i} \in Z_q^*$ at random, computes $H_2(U, \text{ID}_i, M, \text{PK}_{\text{ID}_i}) = h_{2i}\, bP$, then adds them to the list.

H_3 queries: $\mathfrak{C}$ maintains a list H_3^{list}: (U, ID_i, M, PK_{ID_i}, h_{3i}, $h_{3i}P$). When receiving an H_3 query on (U, ID_i, M, PK_{ID_i}), $\mathfrak{C}$ chooses $h_{3i} \in Z_q^*$ at random, computes $H_3(U, \text{ID}_i, M, \text{PK}_{\text{ID}_i}) = h_{3i}P$, and adds them to the list.

Secret value queries: $\mathfrak{A}_{\text{II}}$ requests such a query on ID_i, $\mathfrak{C}$ follows the steps below:

- if the corresponding tuple (ID_i, r_i) already exists in the list, return r_i;
- otherwise,
  - if $i = \tau$, then abort and.
  - if $i \neq \tau$, pick $r_i \in_{\text{R}} Z_q^*$, and add (ID_i, r_i) to the list.

Public key queries: $\mathfrak{A}_{\text{II}}$ requests such a query on ID_i, $\mathfrak{C}$ follows the steps below:

- if the corresponding tuple already exists in the list, return PK_{ID_i};
- otherwise,
  - if $i = \tau$, return $\text{PK}_{\text{ID}_\tau} = aP$ and
  - if $i \neq \tau$, $\mathfrak{C}$ searches the list of secret value for an r_i and computes $\text{PK}_{\text{ID}_i} = r_i P$. If there is not a corresponding tuple with ID_i, $\mathfrak{C}$ picks $r_i \in Z_q^*$ and calculates $\text{PK}_{\text{ID}_i} = r_i P$. Add ($\text{ID}_i$, PK_{ID_i}), (ID_i, r_i) to the public key list and secret value list, respectively.

Sign queries: $\mathfrak{A}_{\text{II}}$ requests such a query on (M, ID), $\mathfrak{C}$ responds as follows:

- if $i \neq \tau$, run the *Sign* algorithm directly and
- else, $\mathfrak{C}$ picks $u \in Z_q^*$ randomly, calculates $U = uP$, $V = S_{\text{ID}^*} + h_{2i}\,\text{PK}_{\text{ID}^*} + h_{3i}P$. The signature is $\sigma = (U, V)$. Note that if a tuple (U, ID, M, PK_{ID}, $\perp$) has already existed, we will pick another $u \in Z_q^*$.

Forge: Finally, $\mathfrak{A}_{II}$ outputs a forgery ($\sigma^* = (U^*, V^*)$, ID^*, M^*). σ^* should satisfy the following verification if it is valid:

$$e(V^*, P) = e(Q_{\text{ID}^*}, P_0)e(h'_2, \text{PK}_{\text{ID}^*})e(h'_3, U^*),$$

where

$$h'_2 = H_2(U^*, \text{ID}^*, M^*, \text{PK}_{\text{ID}^*}),$$

$$h'_3 = H_3(U^*, \text{ID}^*, M^*, \text{PK}_{\text{ID}^*}).$$

Search the H_2^{list} and H_3^{list} for

$$H_2(U^*, \text{ID}^*, M^*, \text{PK}_{ID^*}) = h_{2\tau}bP,$$

$$H_3(U^*, \text{ID}^*, M^*, \text{PK}_{\text{ID}^*}) = h_{3\tau}P,$$

to get $h_{2\tau}$ and $h_{3\tau}$. Obviously, we have

$$\hat{e}(V^* - S_{\text{ID}^*} - h_{3\tau}U^*, P) = \hat{e}(h_{2\tau}abP, P).$$

Now, $\mathfrak{C}$ can easily compute

$$abP = h_{2\tau}^{-1}(V^* - S_{\text{ID}^*} - h_{3\tau}U^*).$$

Analysis: It is not hard for us to gain $\mathfrak{C}$'s advantage $\epsilon' \geq (1/r_{H_2})\epsilon$ in solving the CDH problem within the time

$$t' \leq t + (r_{H_1} + r_{H_2} + r_{H_3} + q_{\text{pk}} + 3q_s)\, t_{\text{mul}}.$$

□

5.2 Aggregate algorithm's security

Theorem 3: In our CLAS scheme, the AS's validity is equivalent to the validity of each individual signature used in the aggregation (Suppose H is a collision resistant Hash function.).

Proof: Suppose each individual signature is valid, then for each $j \in \{1, \ldots, n\}$, we have

$$\hat{e}(V_j, P) = \hat{e}(Q_{\text{ID}_j}, P_0)\hat{e}(h_{2j}, \text{PK}_j)\hat{e}(h_{3j}, U_j).$$

So

$$\hat{e}(V_j, \text{PK}_{\text{sever}}) = \hat{e}(Q_{\text{ID}_j}, sP_0)\hat{e}(h_{2j}, s\text{PK}_j)\hat{e}(h_{3j}, sU_j),$$

and

$$\begin{aligned} h &= H(\hat{e}(V_1, \text{ID}_{\text{sever}}), \ldots, \hat{e}(V_n, \text{PK}_{\text{sever}})) \\ &= H(\hat{e}(Q_{\text{ID}_1}, sP_0)\hat{e}(h_{21}, s\text{PK}_1)\hat{e}(h_{31}, sU_1), \\ &\quad \ldots, \\ &\quad \hat{e}(Q_{\text{ID}_n}, sP_0)\hat{e}(h_{2n}, s\text{PK}_n)\hat{e}(h_{3n}, sU_n)) \\ &= h'. \end{aligned}$$

Therefore

$$\begin{aligned} &\hat{e}(V, P) \\ &= \hat{e}(h\sum_{j=1}^{n} V_j, P) \\ &= \prod_{j=1}^{n} \hat{e}(Q_{\text{ID}_j}, hP_0)\hat{e}(h_{2j}, h\text{PK}_j)\hat{e}(h_{3j}, hU_j) \\ &= \prod_{j=1}^{n} \hat{e}(Q_{\text{ID}_j}, h'P_0)\hat{e}(h_{2j}, h'\text{PK}_j)\hat{e}(h_{3j}, h'U_j). \end{aligned}$$

This indicates that the AS is valid. On the other hand, if

$$\hat{e}(V, P) = \prod_{j=1}^{n} \hat{e}(Q_{\text{ID}_j}, h'P_0)\hat{e}(h_{2j}, h'\text{PK}_j)\hat{e}(h_{3j}, h'U_j),$$

then

$$\hat{e}\left(h\sum_{j=1}^{n}V_j,\ P\right)$$
$$=\prod_{j=1}^{n}\hat{e}\left(Q_{\mathrm{ID}_j},\ hP_0\right)\hat{e}\left(h_{2j},\ h\mathrm{PK}_j\right)\hat{e}\left(h_{3j},\ hU_j\right)$$
$$=\prod_{j=1}^{n}\hat{e}\left(Q_{\mathrm{ID}_j},\ h'P_0\right)\hat{e}\left(h_{2j},\ h'\mathrm{PK}_j\right)\hat{e}\left(h_{3j},\ h'U_j\right).$$

Thus, we have

$$h'=h,$$

then

$$H\big(\hat{e}(Q_{\mathrm{ID}_1},\ sP_0)\hat{e}(h_{21},\ s\mathrm{PK}_1)\hat{e}(h_{31},\ sU_1),$$
$$\ldots,$$
$$\hat{e}(Q_{\mathrm{ID}_n},\ sP_0)\hat{e}(h_{2n},\ s\mathrm{PK}_n)\hat{e}(h_{3n},\ sU_n)\big)$$
$$=H\big(\hat{e}(V_1,\ \mathrm{PK}_{\mathrm{sever}}),\ \ldots,\ \hat{e}(V_n,\ \mathrm{PK}_{\mathrm{sever}})\big),$$

the collision resistance of hash function H implies that, for every $j\in\{1,\ \ldots,\ n\}$, we have

$$\hat{e}\left(V_j,\ \mathrm{PK}_{\mathrm{sever}}\right)=\hat{e}\left(Q_{\mathrm{ID}_j},\ sP_0\right)\hat{e}\left(h_{2j},\ s\mathrm{PK}_j\right)\hat{e}\left(h_{3j},\ sU_j\right).$$

Moreover, hence for every $j\in\{1,\ \ldots,\ n\}$, we have

$$\hat{e}\left(V_j,\ P\right)=\hat{e}\left(Q_{\mathrm{ID}_j},\ P_0\right)\hat{e}\left(h_{2j},\ \mathrm{PK}_j\right)\hat{e}\left(h_{3j},\ U_j\right).$$

This guarantees the validity of every single signature $\sigma_j=\left(U_j,\ V_j\right)$ on a message m_j signed by signer ID_j with the public key PK_j. □

6 Performance analysis

Next, we will have a performance analysis of our CLAS scheme. Denote L_m and L_{ID} be the overall length of $\{m_1,\ m_2,\ \ldots,\ m_n\}$ and $\{\mathrm{ID}_1,\ \mathrm{ID}_2,\ \ldots,\ \mathrm{ID}_n\}$, respectively. Moreover, let $M_{\mathbb{G}}$, $M_{\mathbb{G}_T}$, Exp, P be the calculation cost of a scalar multiplication in $\mathbb{G}$, a multiplication calculation in $\mathbb{G}_T$, an exponentiation operation in $\mathbb{G}_T$ and a pairing operation in $\mathbb{G}_T$, respectively.

6.1 Communication cost

In this section, the comparison of the two situations' communication cost is given in Table 1. The results show that $(n-1)|\mathbb{G}|$ transmission can be decreased in one process of data aggregation; similarly, $(n-1)|\mathbb{G}|$ storage cost can be decreased.

6.2 Efficiency comparison

In this section, the comparison of efficiency about our CLAS scheme with some existing pairing-based schemes is given in Table 2. Our CLAS scheme and Zhang and Zhang scheme [23] just achieve partial aggregation, then both require linear number of pairings, and ours is needed more pairings in aggregate verify process, Xiong *et al.* [22] have a constant number of pairing operations during the aggregate verify; unfortunately, their scheme has some security weaknesses which have been reported in [20, 24, 27].

Furthermore, we conduct the experiment with JPBC 2.0 running on the test machine with an Intel Core i5-4460s at 2.90 GHz and 8 GB of random access memory to learn the operation costs. The software environment on the test machine is Windows 7 with 64 bit, Java JDK1.8, eclipse, JPBC 2.0. We test the aggregate verify time of data consumer between our scheme and Zhang and Zhang [23] and plot the running time in Fig. 2. Table 2 and Fig. 2 indicate that our scheme has a slightly lower efficiency, but stronger security.

Table 1 Communication cost comparison

	Un-aggregate scheme	Aggregate scheme
user→aggregator	$2n\cdot\lvert\mathbb{G}\rvert+L_m+L_{\mathrm{ID}}$	$2n\cdot\lvert\mathbb{G}\rvert+L_m+L_{\mathrm{ID}}$
aggregator→designated verifier	$2n\cdot\lvert\mathbb{G}\rvert+L_m+L_{\mathrm{ID}}$	$(n+1)\cdot\lvert\mathbb{G}\rvert+L_m+L_{\mathrm{ID}}$

Table 2 Some pairing-based CLAS schemes' efficiency comparison

Scheme	Sign	Aggregate verify	Resist coalition attack
Cheng *et al.* [20]	$4M_{\mathbb{G}}$	$3P+2nM_{\mathbb{G}}$	no
Xiong *et al.* [22]	$3M_{\mathbb{G}}$	$3P+2nM_{\mathbb{G}}$	no
Zhang and Zhang [23]	$3M_{\mathbb{G}}$	$(n+3)P+2nM_{\mathbb{G}}+(n+1)M_{\mathbb{G}_T}$	no
ours	$3M_{\mathbb{G}}$	$(2n+1)P+2nM_{\mathbb{G}_T}+2n\,\mathrm{Exp}$	yes

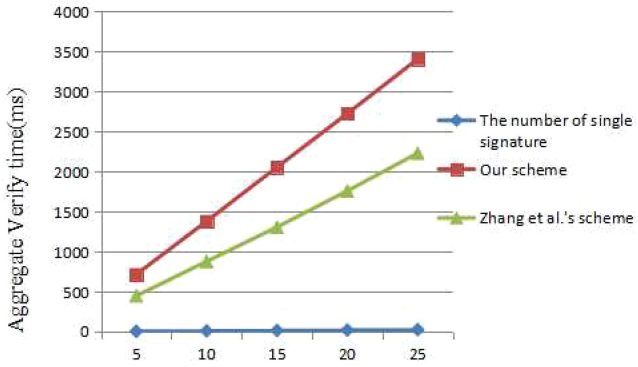

Fig. 2 *Comparison of the aggregate verify time*

7 Conclusion

In this paper, we gave an improved security model for CLAS schemes. The new model attaches importance to the security both of the basic CLS scheme and the aggregate algorithm. Thus, our model can resist a kind of practical and powerful attacks named coalition attacks. We consider this kind of adversary succeeds if it can forge a valid AS utilising several single signatures, while some of the single signatures are invalid. Then, we presented a secure CLAS scheme with a designated verifier. The rigorous security analysis in the random oracle model under the CDH assumption shows that our CLAS scheme is EUF-CMA secure. Furthermore, Theorem 3 indicates that our scheme can resist coalition attacks, i.e. the AS's validity in our scheme is equal to all involved individual signatures' validity. Next, we will be dedicated to designing secure and efficient AS schemes without designated verifiers in various surroundings.

8 Acknowledgments

This work was supported by the National High Technology Research and Development Program (863 Program) of China (No. 2015AA016007), the National Natural Science Foundation of China (Grant nos. 61802195 , 61502237, 61672289, 61572198 and U1405255) and the Shaanxi Science & Technology Coordination & Innovation Project (No. 2016TZC-G-6-3).

9 References

[1] Al-Riyami, S.S., Paterson, K.G.: 'Certificateless public key cryptography'. Proc. ASIACRYPT 2003, Taipei, Taiwan, 30 November–4 December 2003, pp. 452–473

[2] Huang, X., Susilo, W., Mu, Y., *et al.*: 'On the security of certificateless signature schemes from Asiacrypt 2003'. Proc. CANS 2005, Xiamen, Fujian Province, China, 14–16 December 2005, pp. 13–25

[3] Huang, X., Mu, Y., Susilo, W., *et al.*: 'Certificateless signatures: new schemes and security models', *Comput. J.*, 2012, **55**, pp. 457–474

[4] Yu, Y., Mu, Y., Wang, G., *et al.*: 'Improved certificateless signature scheme provably secure in the standard model', *IET Inf. Sec.*, 2012, **6**, (2), pp. 102–110
[5] Shen, L., Zhang, F., Sun, Y.: 'Efficient revocable certificateless encryption secure in the standard model', *Comput. J.*, 2014, **57**, (4), pp. 592–601
[6] Cheng, L., Wen, Q.: 'Cryptanalysis and improvement of a certificateless partially blind signature', *IET Inf. Sec.*, 2015, **9**, (6), pp. 380–386
[7] Sun, Y., Zhang, F., Shen, L., *et al.*: 'Efficient revocable certificateless encryption against decryption key exposure', *IET Inf. Sec.*, 2015, **9**, (3), pp. 158–166
[8] Boneh, D., Gentry, C., Lynn, B., *et al.*: 'Aggregate and verifiably encrypted signatures from bilinear maps'. Proc. EUROCRYPT, Warsaw, Poland, 2003, pp. 416–432
[9] Shen, L., Ma, J., Liu, X., *et al.*: 'A secure and efficient id-based aggregate signature scheme for wireless sensor networks', *IEEE Internet Things J.*, 2017, **4**, (2), pp. 546–554
[10] Tang, J., Liu, A., Zhao, M., *et al.*: 'An aggregate signature based trust routing for data gathering in sensor networks', *Secur. Commun. Netw.*, 2018, **2018**, pp. 1–30, doi.org/10.1155/2018/6328504
[11] Cui, J., Zhang, J., Zhong, H., *et al.*: 'An efficient certificateless aggregate signature without pairings for vehicular *ad hoc* networks', *Inf. Sci.*, 2018, **451–452**, pp. 1–15
[12] Zhang, L., Wu, Q., Domingo-Ferreret, J., *et al.*: 'Distributed aggregate privacy-preserving authentication in VANETs', *IEEE Trans. Intell. Transp. Syst.*, 2017, **18**, (3), pp. 516–526
[13] Bellare, M., Namprempre, C., Neven, G.: 'Unrestricted aggregate signatures'. Proc. Int. Colloquium on Automata, Languages, and Programming, Wroclaw, Poland, July, 2007, pp. 411–422
[14] Li, J., Kim, K., Zhang, F., *et al.*: 'Aggregate proxy signature and verifiably encrypted proxy signature'. Proc. Int. Conf. Provable Security 2007, Wollongong, Australia, November, 2007, pp. 208–217
[15] Shao, Z.H.: 'Enhanced aggregate signatures from pairings'. Proc. CISC, Beijing, China, December 2005, pp. 140–149
[16] Wen, Y., Ma, J., Huang, H.: 'An aggregate signature scheme with specified verifier', *Chin. J. Electron.*, 2011, **20**, (2), pp. 333–336
[17] Gentry, C., Ramzan, Z.: 'Identity-based aggregate signatures'. Proc. PKC 2006, New York, USA, 2006, pp. 257–273
[18] Herranz, J.: 'Deterministic identity-based signatures for partial aggregation', *Comput. J.*, 2006, **49**, (3), pp. 322–330
[19] Shim, K.A.: 'An ID-based aggregate signature scheme with constant pairing computations', *J. Syst. Softw.*, 2010, **83**, (10), pp. 1873–1880
[20] Cheng, L., Wen, Q., Jin, Z., *et al.*: 'Cryptanalysis and improvement of a certificateless aggregate signature scheme', *Inf. Sci.*, 2015, **295**, pp. 337–346
[21] Gong, Z., Long, Y., Hong, X., *et al.*: 'Two certificateless aggregate signatures from bilinear maps'. Proc. SNPD 2007, Qingdao, China, 2007, pp. 188–193
[22] Xiong, H., Guan, Z., Chen, Z., *et al.*: 'An efficient certificateless aggregate signature with constant pairing computations', *Inf. Sci.*, 2013, **219**, pp. 225–235
[23] Zhang, L., Zhang, F.: 'A new certificateless aggregate signature scheme', *Comput. Commun.*, 2009, **32**, (6), pp. 1079–1085
[24] Zhang, F., Shen, L., Wu, G.: 'Notes on the security of certificateless aggregate signature schemes', *Inf. Sci.*, 2014, **287**, pp. 32–37
[25] Viet, N., Ogata, W.: 'Certificateless aggregate signature schemes with improved security', *IEICE Trans. Fundam.*, 2015, **98**, (1), pp. 92–99
[26] Shen, L., Ma, J., Liu, X., *et al.*: 'A provably secure aggregate signature scheme for healthcare wireless sensor networks', *J. Med. Syst.*, 2016, **40**, (11), pp. 1–10, doi:10.1007/s10916-016-0613-3
[27] He, D., Tian, M., Chen, J.: 'Insecurity of an efficient certificateless aggregate signature with constant pairing computations', *Inf. Sci.*, 2014, **268**, pp. 458–462