International Journal of Security and Its Applications
Vol. 10, No. 3 (2016), pp.55-68
http://dx.doi.org/10.14257/ijsia.2016.10.3.06

A secure Certificateless Aggregate Signature Scheme

Baoyuan Kang and Danhui Xu

School of Computer science and software
Tianjin polytechnic university, Tianjin, 300387, China
baoyuankang@aliyun.com , mixueren123123@sina.com

Abstract

Aggregate signatures allow n signatures on n distinct messages from n distinct signers to be aggregated into a single signature that convinces any verifier that n signers do indeed sign the n messages, respectively. The major advantage of utilizing aggregate signatures is to address the security of data and save bandwidth and computations in sensor networks. Recently, people discuss aggregate signature in certificateless public key setting. But some existing certificateless aggregate signature schemes are not secure. In this paper, we analyze the security of Zhang et al.'s certificateless aggregate signature schemes, and propose a new certificateless aggregate signature schemes, and prove the new scheme is existentially unforgeable under adaptive chosen-message attacks under the assumption that computational Diffie–Hellman problem is hard. Furthermore, in signing equation of the proposed scheme user's partial private key and secret value are directly combined with the signed message. So, the scheme is also secure against some inside forgery attack.

Keywords: *Digital Signature; Aggregate Signature; Certificateless aggregate signature; Security; Bilinear Maps*

1. Introduction

To surmount the key escrow problem in identity-based public key cryptography, AI-Riyami and Paterson originated certificateless public key cryptography [1], there also is a key generation center (KGC) to help users to produce private keys. But, KGC only provides the partial private key. The full private key is generated by the user himself through using the partial private key from KGC and the secret information from himself.

Aggregate signature was first invented by Boneh, Centry, Lynn and Shacham [2]. It combines *n* signatures from *n* different signers on *n* different messages into a single small-size signature. The single small-size signature is used to examine whether the *n* signers did sign the *n* messages, respectively. Aggregate signatures are useful in secure routing [7] and certificate chain compression [2]. The primary advantage of aggregate signature is in saving bandwidth. Such aggregate signature is a good solution for networks of small, battery-powered devices where communication over energy-consuming wireless sensor networks channels [9]. For instance, in unattended wireless sensor network, the lack of real-time communication and resource constraints bring security and performance challenges [19]. To prevent an attacker from altering the data accumulated, forward secure signature [20] is used to sign the accumulated data. But this also causes storage and communication overheads due to the accumulation of the signature of individual data items. Aggregate signature scheme may be developed to address this issue by aggregate the signature of different data items to a single small-size signature.

After the first aggregate signature scheme proposed by Boneh and colleages, a number of aggregate signature schemes were proposed [3, 4, 8, 10, 11, 12, 13]. Recently, people discuss aggregate signature in certificateless public key setting. Some certificateless aggregate signature (CLAS) schemes are proposed [5, 14, 15, 16]. Xiong et al. [14]

contributed a certificateless aggregate signature with constant pairing computations. However, He et al. [6] showed that an adversary in Xiong et al. scheme who knows master key could forge a valid certificateless aggregate signature for any message under any identity. In [16] Zhang et al. proposed a new certificateless aggregate signature scheme. But in the signing equation of the scheme, signer's partial private key and secret value are separated from the signed message. So, Zhang et al.'s scheme is vulnerable to inside attack. The two certificateless aggregate signature schemes [5] proposed by Gong et al. are suffer from same flaws of Zhang et al.'s scheme. Therefore, it is necessary to construct novel certificateless aggregate signature scheme which is provably secure. In this paper, we propose an improved scheme based on Zhang et al.'s scheme [16], and prove that the proposed scheme is existentially unforgeable under adaptive chosen-message attacks under the assumption of the computational Diffie-Hellman problem being hard. Furthermore, in signing equation of the proposed scheme signer's partial private key and secret value are directly combined with the signed message. So, the proposed scheme is also secure against some potential inside forgery attack. In the era of big data it is vital of data security [21], the secure certificateless aggregate signature (CLAS) schemes are important to dig data.

The rest of the paper is organized as follows. Section 2 introduces cryptographic hardness assumptions and the definition and security model of certificateless aggregate signature. Section 3 reviews Zhang et al.'s certificateless aggregate signature scheme and shows that Zhang et al.'s scheme is vulnerable to inside attack. The improved certificateless aggregate signature scheme is proposed in Section 4. Section 5 discusses the security of improved scheme. Finally, Section 6 concludes this paper.

2. Preliminaries

2.1 Bilinear Maps and Complexity Assumption

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

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

(2) Non-degeneracy: There exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.

(3) Computable: 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 cyclic group G with order q, and given (aP, bP) for unknown $a, b \in Z_q^*$, to compute abP.

2.2 Definition of Certificateless Aggregate Signature Schemes

In accordance with [16], A certificateless aggregate signature scheme includes a KGC, an aggregating set U of n users $U_1, \cdots, U_n$ and an aggregate signature generator. There are six algorithms: Setup, Partial-private-key-extract, userkeygen, sign, aggregate and aggregate verify.

Setup: This algorithm run by KGC, given a security parameter τ, outputs a master key and a list of system parameters params.

Partial-private-key-extract: This algorithm executed by KGC, given a user's identity

ID_i, a parameter list params and a master-key, outputs the user's partial private key D_i.

Userkeygen: An algorithm run by a user, given the user's identity ID_i, picks a random $x_i \in Z_q^*$ and produces the user's secret value/public key x_i / P_i.

Sign: An algorithm executed by each user U_i in an aggregating set U. Its inputs are the parameter list params, a message $m_i \in \{0,1\}^*$, U_i's identity ID_i, his public key P_i and his signing key (x_i, D_i). The output is a signature σ_i on message m_i, which is valid under U_i's identity ID_i and his public key P_i.

Aggregate: An algorithm run by the aggregate signature generator, given an aggregate set set U of n users $U_1, \cdots, U_n$, the identity ID_i of each U_i, the public key P_i of U_i, and a signature σ_i on a message m_i under identity ID_i and public key P_i for each user $U_i \in U$, outputs an aggregate signature σ on messages $m_1, \cdots, m_n$.

Aggregate verify: Given an aggregate set U of n users $U_1, \cdots, U_n$, the identity ID_i of each U_i, the public key P_i of U_i, an aggregate signature σ on messages $m_1, \cdots, m_n$, the algorithm outputs true if the aggregate signature is valid, or false otherwise.

2.3 Security Model of Certificateless Aggregate Signature Schemes

Based on the security model of certificateless signature scheme [1], there are two types adversaries A_1 and A_2 for CLAS. Type 1 adversary A_1 does not have access to the master key, but he can replace the public key of any user. While type 2 adversary A_2 has access to the master-key but cannot perform public key replacement.

The security of CLAS scheme is modeled in the following two games between a challenger S and an adversary A_1 or A_2.

Game 1 (For Type 1 Adversary)

Setup: The challenger S executes this algorithm, given a security parameter τ, the algorithm outputs a master key and the system parameter list params. Then, S sends params to the adversary A_1 while holds the master key secret.

Attack: The adversary A_1 executes a polynomial bounded number of queries in an adaptive manner.

Partial-private-key-queries: When A_1 queries the partial private key of a user with identity ID_i, S outputs the partial private key D_i for the user.

Public-key queries: When A_1 queries the public key of whose identity is ID_i, S outputs it.

Secret-Value queries: When A_1 queries the secret value of a user whose identity is ID_i, S outputs the secret value x_i (It outputs $\perp$, if the user's public key has been replaced).

Public-key-replacement queries: For any user with identity ID_i. A_1 can select a new public key P_i' for him and record this replacement.

Sign queries: When A_1 queries a user's (whose identity is ID_i) signature on a message m_i, S generates a valid signature σ_i on message m_i under identity ID_i and public key

P_i.

Forgery: A_1 outputs a set of n users $U^* = \{U_1^*, \cdots, U_n^*\}$ with identities set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$ and corresponding public keys set $L_{PK}^* = \{P_1^*, \cdots, P_n^*\}$, n message set $L_M^* = \{m_1^*, \cdots, m_n^*\}$, and an aggregate signature σ^*.

A_1 succeeds if and only if

(1) σ^* is a valid aggregate signature on message $m_1^*, \cdots, m_n^*$ under identities $ID_1^*, \cdots, ID_n^*$ and the corresponding public keys $P_1^*, \cdots, P_n^*$ chosen by A_1.

(2) At least one identity, let be $ID_1^* \in L_{ID}^*$, has not been submitted during the partial-private key queries and the signature on m_1^* under ID_1^* and the corresponding public key P_1^* has never been queried during the sign queries. Here P_1^* denotes the public key of the user U_1^* whose identity is ID_1^*.

Game 2 (For Type 2 Adversary)

Setup: Given a security parameter τ, S runs the algorithm to obtain a master-key and the system parameter list params. Then S sends params and the master-key to the adversary A_2.

Attack: The adversary A_2 executes a polynomial bounded number of queries in an adaptive manner.

Public-key queries: When A_2 queries the public key of a user (whose identity is ID_i) of his choice. S outputs the public key P_i for this user.

Secret-Value queries: When A_2 queries a user's (whose identity is ID_i) secret value, S outputs the secret value x_i for the user.

Sign queries: When A_2 queries a user's (whose identity is ID_i) signature on a message m_i, S replies with a signature σ_i on message m_i under identity ID_i and public key P_i.

Forgery: A_2 outputs a set of n users $U^* = \{U_1^*, \cdots, U_n^*\}$ with identities set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$, public keys set $L_{PK}^* = \{P_1^*, \cdots, P_n^*\}$, n message set $L_M^* = \{m_1^*, \cdots, m_n^*\}$, and an aggregate signature σ^*.

A_2 succeeds, if and only if

(1) σ^* is a valid aggregate signature on message $m_1^*, \cdots, m_n^*$ under identities $ID_1^*, \cdots, ID_n^*$ and the corresponding public keys form the set $P_1^*, \cdots, P_n^*$ chosen by A_2.

(2) At least one identity, let be $ID_1^* \in L_{ID}^*$, has not been submitted during the secret-value queries. The signature on m_1^* under ID_1^* and the corresponding public key P_1^* has never been queried during the sign queries.

Definition A CLAS scheme is existentially unforgeable under adaptive chosen-message attack if and only if the success probability of any polynomial bounded adversary in any of the above two games is negligible.

3. The Security of Zhang *et al*. Certificateless Aggregate Signature Scheme

In this section we analyze the security of Zhang *et al*.'s certificateless aggregate signature scheme. Zhang *et al*.'s scheme is a typical one that is vulnerable to inside forgery attack.

3.1 Brief review of Zhang *et al*.'s Scheme

In [16], Zhang *et al*. proposed an efficient certificateless aggregate signature scheme that consists of the following six algorithms.

Setup: Given a security parameter l, the KGC chooses a cyclic additive group G_1 which is generated by P with prime order q, chooses a cyclic multiplicative group G_2 of the same order and a bilinear map $e{:}G_1 \times G_1 \to G_2$. The key generation center (KGC) also chooses a random $\lambda \in Z_q^*$ as the master-key and sets $P_T = \lambda P$, chooses cryptographic hash functions $H_1{:}\{0,1\}^* \to G_1$, $H_2{:}\{0,1\}^* \to G_1$, $H_3{:}\{0,1\}^* \to G_1$. The system parameter list is params$=(G_1,G_2,e,P,P_T,H_1,H_2,H_3)$. The message space is $\mu=\{0,1\}^*$.

Partial-Private-Key-Extract: This algorithm accepts params, master-key λ and a user's identity $ID_i \in \{0,1\}^*$. It generates the partial private key for the user as follows:

(1) Computes $Q_i = H_1(ID_i)$.

(2) Outputs the partial private key $D_i = \lambda Q_i$.

UserKeyGen: This algorithm takes as input a user's identity ID_i, selects a random $x_i \in Z_q^*$ and sets his secret value/public key as $x_i / P_i = x_i P$.

Sign: To sign a message $M_i \in \mu$ using the signing key (x_i, D_i), the signer, whose identity is ID_i and the corresponding public key is P_i, first chooses a state information Δ (for our scheme, we can choose some elements of the system parameters as Δ), then performs the following steps:

(1) Chooses a random $r_i \in Z_q^*$, compute $R_i = r_i P$.

(2) Computes $W = H_2(\Delta)$, $S_i = H_3(\Delta \| M_i \| ID_i \| P_i \| R_i)$.

(3) Computes $V_i = D_i + x_i W + r_i S_i$.

(4) Outputs $\sigma_i = (R_i, V_i)$ as the signature on M_i.

Aggregate: Anyone can act as an aggregate signature generator who can aggregate a collection of individual signatures that use the same state information Δ. For an aggregating set U (which has the same state information Δ) of n users $U_1,...,U_n$ with identities $ID_1,...,ID_n$ and the corresponding public keys $P_1,...,P_n$ and message-signature pairs $(M_1,\sigma_1=(R_1,V_1)),...,(M_n,\sigma_n=(R_n,V_n))$ from $U_1,...,U_n$, respectively, the aggregate signature generator computes $V=\sum_{i=1}^{n} V_i$ and outputs the aggregate signature $\sigma=(R_1,...,R_n,V)$.

Aggregate Verify: To verify an aggregate signature $\sigma=(R_1,...,R_n,V)$ signed by n user

$U_1,...,U_n$ with identities $ID_1,...,ID_n$ and corresponding public keys $P_1,...,P_n$, on messages $M_1,...,M_n$ with the same string Δ, the verifier performs the following steps:

(1) Computes $W = H_2(\Delta)$, and for all $i, 1 \le i \le n$ compute $Q_i = H_1(ID_i)$, $S_1 = H_3(\Delta \| M_i \| ID_i \| P_i \| R_i)$.

(2) Verifies $e(V,P) \stackrel{?}{=} e(P_T, \sum_{i=1}^{n} Q_i) e(W, \sum_{i=1}^{n} P_i) \prod_{i=1}^{n} e(S_i, R_i)$.

(3) If the equation holds, outputs $true$. Otherwise, outputs $false$.

3.2 Attack on Zhang *et al*.'s Scheme

According to the concept of inside attack on aggregate signature proposed in [17, 18], there is an inside attack on Zhang *et al*.'s scheme.

Let ID_1 and ID_2 be identities of two signers U_1, ID_2, respectively. U_1 and U_2 declare that they produce a certificateless aggregate signature $\sigma = (R_1, R_2, V)$ on messages (M_1, M_2) for (ID_1, ID_2). In the light of the design of Zhang *et al*.'s scheme, Of course, U_1 and U_2 should sign M_1, M_2, respectively. But, U_1 and U_2 may maliciously do as following:

1. U_1 and U_2 Choose $r_1, r_2 \in_R Z_q^*$ and compute $R_1 = r_1 \cdot P$ and $R_2 = r_2 \cdot P$,
2. U_1 and U_2 cooperate to compute

$$W = H_2(\Delta),\ S_1 = H_3(\Delta \| M_1 \| ID_1 \| P_1 \| R_1),\ S_2 = H_3(\Delta \| M_2 \| ID_2 \| P_2 \| R_2)$$

$$V_1^* = D_1 + x_1 W + r_2 S_2,\ V_2^* = D_2 + x_2 W + r_1 S_1.$$

Note they have not signed M_1 and M_2, respectively.

3. They declare that they generate certificateless aggregate signature $\sigma = (R_1, R_2, V^*)$ on messages (M_1, M_2) for (ID_1, ID_2). Here $V^* = V_1^* + V_2^*$.

In fact, $V^* = V_1^* + V_2^* = V_1 + V_2 = V$. So, the verification equation

$$e(V^*, P) = e(P_T, Q_1 + Q_2) e(W, P_1 + P_2) e(S_1, R_1) e(S_2, R_2)$$

holds, U_1 and U_2 successfully forge an aggregate signature for (ID_1, ID_2) on (M_1, M_2). So, Zhang *et al*.'s scheme is subjected to inside forgery attack.

4. A New Certificateless Aggregate Signature Scheme

To overcome the flaw of Zhang *et al*. scheme [16], in this section we construct a new CLAS scheme which can be regard as an improvement of Zhang *et al*. scheme. The new scheme consists of six algorithms, Setup, Partial-private-key-extract, UserKeyGen, Sign, Aggregate, Aggregate Verify. KGC performs Setup algorithm and Partial-private-key-extract algorithm to generate the system parameter and user's partial private key. n distinct signers generate n signatures on n distinct messages. On obtaining the n signatures, anyone can act as an aggregate signature generator to aggregate the n individual signatures into a single signature.

Setup: Given a security parameter λ, the KGC chooses a cyclic additive group G_1 which is generated by P and a cyclic multiplicative group G_2. G_1 and G_2 have the same order q. Then, KGC chooses a bilinear map $e: G_1 \times G_1 \to G_2$ and random $s \in Z_q^*$ as the master key and sets system public $P_{pub} = sP$, picks four cryptographic hash functions

$H_1:\{0,1\}^* \to G_1$, $H_2:\{0,1\}^* \to Z_q$, $H_3:\{0,1\}^* \to G_1$, $H_4:\{0,1\}^* \to G_1$.

The system parameter list is $params = (G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4)$. The message space is $\mu=\{0,1\}^*$.

Partial-Private-Key-Extract: Given master key s and a user's identity $ID_i \in \{0,1\}^*$, this algorithm generates the partial private key for the user as follows:

(1) Computes $Q_i = H_1(ID_i)$.

(2) Outputs the partial private key $D_i = sQ_i$.

UserKeyGen: Given a user's identity ID_i, This algorithm picks a random $x_i \in Z_q^*$ as the user's secret value, and computes his public key as $P_i = x_i P$.

Sign: After choosing a state information w, A signer with identity ID_i and public key P_i, signs a message $m_i \in \mu$ using the signing key (x_i, D_i) by the following steps:

(1) Chooses a random $r_i \in Z_q^*$, computes $R_i = r_i P$.

(2) Computes $h_{2i} = H_2(m_i, ID_i, P_i, R_i)$, $H_{3i} = H_3(m_i, ID_i, P_i, R_i)$, $T = H_4(w)$

(3) Computes $V_i = h_{2i} D_i + x_i H_{3i} + r_i T$.

(4) Outputs $\sigma_i = (R_i, V_i)$ as the signature on m_i.

Aggregate: For the same state information w and an aggregating set U (which has the same state information w) of n users $U_1, \cdots, U_n$ with identities $ID_1, \cdots, ID_n$ and the corresponding public keys $P_1, \cdots, P_n$ and message-signature pairs $((m_1, \sigma_1 = (R_1, V_1)), \cdots, (m_n, \sigma_n = (R_n, V_n))$ from $U_1, \cdots, U_n$, respectively, any aggregate signature generator can computes $V = \sum_{i=1}^{n} V_i$ and outputs the aggregate signature $\sigma = (R_1, \cdots, R_n, V)$.

Aggregate Verify: To verify an aggregate signature $\sigma = (R_1, \cdots, R_n, V)$ signed by n user $U_1, \cdots, U_n$ with identities $ID_1, \cdots, ID_n$ and corresponding public keys $P_1, \cdots, P_n$, on messages $m_1, \cdots, \mathrm{m}_n$, the verifier executes the following steps:

(1) Computes $Q_i = H_1(ID_i)$, $h_{2i} = H_2(m_i, ID_i, P_i, R_i)$, $H_{3i} = H_3(m_i, ID_i, P_i, R_i)$, for all $i, 1 \le i \le n$, and $T = H_4(w)$.

(2) Verifies $e(V, P) \overset{?}{=} e(T, \sum_{i=1}^{n} R_i) e(P_{pub}, \sum_{i=1}^{n} h_{2i} Q_i) \prod_{i=1}^{n} e(H_{3i}, P_i)$.

(3) If the equation holds, outputs $true$. Otherwise, outputs $false$.

5. Security Proof

Theorem 1. In the random oracle model, our certificateless aggregate scheme is existentially unforgeable against type 1 adversary under the assumption that the CDH problem in G_1 is intractable.

Proof. Let S_1 be a CDH attacker who receives a random instance (P, aP, bP) of the CDH problem in G_1. A_1 a type 1 adversary who interacts with S_1 as modeled in Game 1. We show how S_1 may use A_1 to solve the CDH problem, *i.e.* to compute abP.

Setup: Firstly, S_1 sets system public $P_{pub} = aP$ and selects $params = (G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4)$.

Then he sends params to A_1.

Attack: A_1 performs following types of queries in an adaptive manner and we consider hash functions H_1, H_2, H_3 and H_4 as random oracles.

H_1 queries: S_1 keeps a list H_1^{list} of tuples $(ID_j, \alpha_j, Q_j, c_j)$. On receiving a H_1 query on ID_i the same answer will be given if the query has been recorded on the list H_1^{list}. Otherwise, S_1 selects $\alpha_i \in Z_q^*$ at random and flips a coin $c_i \in \{0,1\}$. If $c_i = 0$, S_1 sets $Q_i = \alpha_i bP$, adds $(ID_i, \perp, Q_i, c_i)$ to H_1^{list} and returns Q_i as answer. Otherwise, sets $Q_i = \alpha_i P$, adds $(ID_i, \alpha_i, Q_i, c_i)$ to H_1^{list} and returns Q_i as answer.

H_2 queries: S_1 keeps a list H_2^{list} of tuples $(m_i, ID_i, P_i, R_i, \beta_i)$. Whenever A_1 queries $H_2(m_i, ID_i, P_i, R_i)$, the same answer will be given if the query has been recorded on the list H_2^{list}. Otherwise, S_1 selects a random $\beta_i \in Z_q^*$, adds $(m_i, ID_i, P_i, R_i, \beta_i)$ to H_2^{list} and returns β_i as answer.

H_3 queries: S_1 keeps a list H_3^{list} of tuples $(m_i, ID_i, P_i, R_i, \lambda_i, \gamma_i)$. When A_1 queries $H_3(m_i, ID_i, P_i, R_i)$, the same answer will be given if the query has been recorded on the list H_3^{list}. Otherwise, S_1 picks a random $\gamma_i \in Z_q^*$, computes $\lambda_i = \gamma_i P$, adds $(m_i, ID_i, P_i, R_i, \lambda_i . \gamma_i)$ to H_3^{list} and returns λ_i as answer.

H_4 queries: S_1 keeps a list H_4^{list} of tuples (w_i, T_i, ζ_i).This list is initially empty. Whenever A_1 issues a query $H_4(w_i, T_i, \zeta_i)$, the same answer will be given if the query has been recorded on the list H_4^{list}. Otherwise, S_1 selects a random $\zeta_i \in Z_q^*$, computes $T_i = \zeta_i P$, adds (w_i, T_i, ζ_i) to H_4^{list} and returns T_i as answer.

Partial-Private-Key queries: S_1 maintains a list K^{list} of tuples (ID_j, x_j, D_j, P_i). When A_1 queries a Partial-Private-Key for ID_i, the same answer from the list K^{list} will be given if the request has been asked before. Otherwise, S_1 first does an H_1 query on ID_i and finds the tuple $(ID_j, \alpha_j, Q_j, c_j)$ on H_1^{list}, then does as follows:

(1) If $c_i = 0$, abort.

(2) Else if there's a tuple (ID_j, x_j, D_j, P_i) on K^{list}, set $D_i = \alpha_i P_{pub}$, and return D_i as answer.

(3) Otherwise, compute $D_i = \alpha_i P_{pub}$, set $x_i = P_i = \perp$, then return D_i as answer and add (ID_j, x_j, D_j, P_i) on K^{list}.

Public-Key queries: On receiving a Public-Key query on ID_i, if the request has been recorded on the list K^{list}, the same answer will be given. Otherwise, S_1 does as follows:

(1) If there's a tuple (ID_j, x_j, D_j, P_i) on K^{list} (in this case, the public key P_i of ID_i is

$\perp$), choose $x_i^{'} \in Z_q^*$, compute $P_i^{'} = x_i^{'}P$, return $P_i^{'}$ as answer and update (ID_j, x_j, D_j, P_i) to $(ID_j, x_j^{'}, D_j, P_j^{'})$.

(2) Otherwise, choose $x_i \in Z_q^*$, compute $P_i = x_i P$,return P_i as answer, set $D_i = \perp$ and add (ID_i, x_i, D_i, P_i) to K^{list}.

Secret-Value queries: On receiving a Secret-Value query on ID_i , S_1 first makes public-key query on ID_i , then finds the tuple (ID_i, x_i, D_i, P_i) on K^{list} and returns x_i as answer (Note that the value of x_i maybe $\perp$).

Public-Key-Replacement queries: A_1 can choose a new public key for the user whose identity is ID_i. On receiving a Public-Key-Replacement query, S_1 first finds the tuple (ID_i, x_i, D_i, P_i) on K^{list} (if such a tuple does not exists on K^{list} or $P_i = \perp$, S_1 first makes Public-Key query on ID_i, then S_1 updates P_i to $P_i^{'}$.

Sign queries: On receives a Sign query on m_i by user with identity ID_i . S_1 first recovers $(ID_i, \alpha_i, Q_i, c_i)$ from H_1^{list}, $(m_i, ID_i, P_i, R_i, \beta_i)$ from H_2^{list} and then generates the signature as follows:

(1) If $c_i = 0$, chooses $t_i, d_i \in Z_q^*$, sets $R_i = t_i^{-1}(d_i P - P_i - P_{pub})$, $H_{3i} = \beta_i Q_i$, $T = t_i \beta_i Q_i$ and adds $(m_i, ID_i, P_i, R_i, \beta_i, \beta_i Q_i)$ on H_3^{list}, and (w_i, T_i, t_i) on H_4^{list} computes $V_i = d_i \beta_i Q_i$, outputs $\sigma_i = (R_i, V_i)$. Here β_i is recovered in $(m_i, ID_i, P_i, R_i, \beta_i)$ from H_2^{list}.

(2) If $c_i = 1$, randomly chooses $R_i \in G_1$, sets $V_i = \alpha_i \beta_i P_{pub} + \gamma_i P_i + \zeta_i R_i$, outputs $\sigma_i = (R_i, V_i)$. Here γ_i is recovered from $(m_i, ID_i, P_i, R_i, \lambda_1 . \gamma_i)$ on H_3^{list}, ζ_i is recovered from (w_i, T_i, ζ_i) on H_4^{list}

Forgery: Eventually, A_1 returns a set U of n users, whose identities form the set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$ and the corresponding public keys form the set $L_{PK}^* = \{P_1^*, \cdots, P_n^*\}$, n messages form the set $L_m^* = \{m_1^*, \cdots, m_n^*\}$, a forged aggregate signature $\sigma^* = \{R_1^*, \cdots, R_n^*, V^*\}$. It is required that there exists $I \in \{1, \cdots, n\}$ such that A_1 has not asked the partial private key for ID_I. And A_1 has not made sign query on (m_I^*, ID_I^*, P_I^*). Without loss of generality, we let $I = 1$. The forged aggregate signature must satisfy

$$e(V^*, P) = e(T^*, \sum_{i=1}^n R_i^*) e(P_{pub}, \sum_{i=1}^n h_{2i}^* Q_i^*) \prod_{i=1}^n e(H_{3i}^*, P_i^*) .$$

Where

$Q_i^* = H_1(ID_i^*)$, $h_{2i}^* = H_2(m_i^*, ID_i^*, P_i^*, R_i^*)$, $H_{3i}^* = H_3(m_i^*, ID_i^*, P_i^*, R_i^*)$, $T^* = H_4(\mathrm{w}^*)$.

S_1 recovers $(ID_i^*, \alpha_i^*, Q_i^*, c_i^*)$ from H_1^{list} , $(m_i^*, ID_i^*, P_i^*, R_i^*, \beta_i^*)$ from H_2^{list} , $(m_i^*, ID_i^*, P_i^*, R_i^*, \lambda_i^*, \gamma_i^*)$ from H_3^{list} and $(w_i^*, T_i^*, \zeta_i^*)$ from H_4^{list} for all $i, 1 \le i \le n$. S_1 proceeds only if $c_1^* = 0$, $c_i^* = 1$ for all $i, 2 \le i \le n$. Otherwise, S_1 aborts.

Because the forged certificateless aggregate signature must satisfies

$$e(V^*, P) = e(T^*, \sum_{i=1}^n R_i^*) e(P_{pub}, \sum_{i=1}^n h_{2i}^* Q_i^*) \prod_{i=1}^n e(H_{3i}^*, P_i^*)$$

We have that

$$e(P_{pub}, h_{21}^*Q_1^*) = e(V^*, P)(e(T^*, \sum_{i=1}^n R_i^*)e(P_{pub}, \sum_{i=2}^n h_{2i}^*Q_i^*)\prod_{i=1}^n e(h_{3i}^*, P_i^*))^{-1}$$

And by our setting, $Q_1^* = \alpha_1^* bP$, $h_{21}^* = \beta_1^*$, $h_{31}^* = \gamma_1^* P$, $T^* = \zeta^* P$, and for all $i, 2 \le i \le n$, $Q_i^* = \alpha_i^* P$, $h_{2i}^* = \beta_i^*$, $h_{3i}^* = \gamma_i^* P$. So, S_1 can compute

$$abP = (\beta_1^*\alpha_1^*)^{-1}(V^* - \sum_{i=2}^n \beta_i^*\alpha_i^* P_{pub} - \sum_{i=1}^n (\gamma_i P_i^* + \zeta^* R_i^*)).$$

Theorem 2. In the random oracle model, our certificateless aggregate scheme is existentially unforgeable against type 2 adversary under the assumption that the CDH problem in G_1 is intractable.

Proof. Let S_2 be a CDH attacker who receives a random instance (P, aP, bP) of the CDH problem in G_1, and has to compute the value of abP. A_2 a type 2 adversary who interacts with S_2 as modeled in Game 2. We show how S_2 may use A_2 to solve the CDH problem, *i.e* to compute abP.

Setup: Firstly, S_2 selects a random $\eta \in Z_q^*$ as the master-key, computes system public key $P_{pub} = \eta P$. Then selects the system parameters

$$params = (G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4).$$

Then he sends params and the master key η to A_2. Since A_2 has access to the master-key, he can do Partial-Private-key-Extract himself.

Attack: A_2 can perform the following types of queries in an adaptive manner and we need not model the hash functions H_1 as a random oracle in this case.

H_2 queries: S_2 maintains a list H_2^{list} of tuples $(m_i, ID_i, P_i, R_i, \beta_i)$. When A_2 queries $H_2(m_i, ID_i, P_i, R_i)$, the same answer will be given if the query has been recorded on the list H_2^{list}. Otherwise, S_2 selects a random $\beta_i \in Z_q^*$, and adds $(m_i, ID_i, P_i, R_i, \beta_i)$ to H_2^{list}, returns β_i as answer.

H_3 queries: S_2 maintains a list H_3^{list} of tuples $(m_i, ID_i, P_i, R_i, \lambda_i, \gamma_i)$. When A_2 queries $H_3(m_i, ID_i, P_i, R_i)$, the same answer will be given if the query has been recorded on the list H_3^{list} Otherwise, S_2 selects a random $\gamma_i \in Z_q^*$, computes $\lambda_i = \gamma_i aP$, adds $(m_i, ID_i, P_i, R_i, \lambda_i.\gamma_i)$ to H_3^{list} and returns λ_i as answer.

H_4 queries: S_2 maintains a list H_4^{list} of tuples (w_i, T_i, ζ_i). When A_2 issues a query $H_4(w_i, T_i, \zeta_i)$, the same answer will be given if the query has been recorded on the list H_4^{list}. Otherwise, S_2 selects a random $\zeta_i \in Z_q^*$, computes $T_i = \zeta_i P$, adds (w_i, T_i, ζ_i) to H_4^{list} and returns T_i as answer.

Public-Key queries: On receiving a Public-Key query on ID_i, if the request has been recorded on the list K^{list}, the same answer will be given. Otherwise, S selects $x_i \in Z_q^*$ and flips a coin $c_i \in \{0,1\}$. If $c_i = 0$, S returns $x_i bP$, adds (ID_i, P_i, c_i) to K^{list}. Otherwise, computes $P_i = x_i P$, and adds (ID_i, x_i, P_i, c_i) to K^{list} and returns P_i as answer.

Secret-Value queries: On receiving a Secret-Value query on ID_i, S first finds the tuple (ID_i, x_i, P_i, c_i) on K^{list}. If $c_i = 0$, S aborts, otherwise, returns x_i as answer.

Sign queries: On receives a Sign query on m_i by user with identity ID_i. S_2 first recovers (ID_i, x_i, P_i, c_i) on K^{list}, $(m_i, ID_i, P_i, R_i, \beta_i)$ from H_2^{list}, $(m_i, ID_i, P_i, R_i, \lambda_i.\gamma_i)$ from H_3^{list}, and (w_i, T_i, ζ_i) from H_4^{list}, then generates the signature as follows:

(1) If $c_i = 0$, randomly chooses $R_i \in G_1$, and $d_i \in Z_q^*$, set $H_{3i} = d_i P$, adds $(m_i, ID_i, P_i, R_i, d_i, d_i P)$ on H_3^{list}, and computes $V_i = \eta\beta_i H_i(ID_i) + d_i x_i(bP) + \zeta_i R_i$, outputs $\sigma_i = (R_i, V_i)$.

(2) If $c_i = 1$, uses the standard sign algorithm to generate $\sigma_i = (R_i, V_i)$ (because S knows the full signing key of the user whose identity is ID_i), outputs $\sigma_i = (R_i, V_i)$.

Forgery: Eventually, A_1 returns a set U of n users, whose identities form the set $L_{ID}^* = \{ID_1^*, \cdots, ID_n^*\}$ and the corresponding public keys form the set $L_{PK}^* = \{P_1^*, \cdots, P_n^*\}$, n messages form the set $L_m^* = \{m_1^*, \cdots, m_n^*\}$, a forged aggregate signature $\sigma^* = \{R_1^*, \cdots, R_n^*, V^*\}$. It is required that there exists $I \in \{1, \cdots, n\}$ such that A_1 has not asked the partial private key for ID_I. And A_1 has not made sign query on (m_I^*, ID_I^*, P_I^*). Without loss of generality, we let $I = 1$. In addition, the forged aggregate signature must satisfy

$$e(V^*, P) = e(T^*, \sum_{i=1}^{n} R_i^*) e(P_{pub}, \sum_{i=1}^{n} h_{2i}^* Q_i^*) \prod_{i=1}^{n} e(H_{3i}^*, P_i^*).$$

Where

$Q_i^* = H_1(ID_1^*)$, $h_{2i}^* = H_2(m_i^*, ID_i^*, P_i^*, R_i^*)$, $H_{3i}^* = H_3(m_i^*, ID_i^*, P_i^*, R_i^*)$, $T^* = H_4(\mathrm{w}^*)$.

S_2 recovers $(m_i^*, ID_i^*, P_i^*, R_i^*, \beta_i^*)$ from H_2^{list}, $(m_i^*, ID_i^*, P_i^*, R_i^*, \lambda_i^*, \gamma_i^*)$ from H_3^{list} and (w_i, T_i, ζ_i) from H_4^{list} for all $i, 1 \le i \le n$. S_2 proceeds only if $c_1^* = 0$, $c_i^* = 1$ for all $i, 2 \le i \le n$. Otherwise, S_2 aborts.

Because the forged certificateless aggregate signature must satisfies

$$e(V^*, P) = e(T^*, \sum_{i=1}^{n} R_i^*) e(P_{pub}, \sum_{i=1}^{n} h_{2i}^* Q_i^*) \prod_{i=1}^{n} e(H_{3i}^*, P_i^*)$$

We have that

$$e(h_{31}^*, P_1^*) = e(V^*, P)(e(T^*, \sum_{i=1}^{n} R_i^*) e(P_{pub}, \sum_{i=1}^{n} h_{2i}^* Q_i^*) \prod_{i=2}^{n} e(h_{3i}^*, P_i^*))^{-1}$$

And by our setting, $h_{21}^* = \beta_1^*$, $h_{31}^* = \gamma_1^* aP$, $P_1^* = x_1^* bP$, $T^* = \zeta^* P$, and for all $i, 2 \le i \le n$, $h_{2i}^* = \beta_i^*$, $h_{3i}^* = \gamma_i^* aP$, $P_i^* = x_i^* P$. Hence, S_2 can compute

$$abP = (\gamma_1^* x_1^*)^{-1} (V^* - \sum_{i=1}^{n} (\eta\beta_i^* Q_i^* + \zeta^* R_i^*) - \sum_{i=2}^{n} x_i^* \gamma_i^* (aP))..$$

Note Our certificateless aggregate scheme is secure against inside attack. Due to separation of user's partial private key and secret value with messages, Zhang *et al.*'s scheme cannot withstand inside forgery attack. But, in signing equation of the new scheme, user's partial private key and secret value are directly combined with the signed message. So, the new scheme is secure against inside forgery attack. In sensor networks, where with hostile sensors, signature scheme against inside forgery attack is vital to protect data.

6. Conclusion

In this paper, we analyze the security of Zhang *et al*.'s certificateless aggregate signature scheme, give an improved certificateless aggregate signature scheme, and prove that the new scheme is existentially unforgeable under adaptive chosen-message attacks assuming the computational Diffie-Hellman problem is hard. Furthermore, in signing equation of the new scheme, user's partial private key and secret value are directly combined with the signed message. So, the new scheme is also secure against some inside forgery attack. The new scheme may have applications where many different certificateless signatures need to be compressed into one single small-size signature.

Acknowledgements

This work was supported by the Research Programs of Applied Basic and Advanced Technology of Tianjin (No. 15JCYBJC15900).

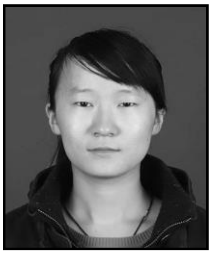
References

[1] S. AI-Riyami, K. Paterson, Certificateless public key cryptography. In: ASIACRYPT'03, LNCS 2894, pp. 452-473. Springer, Heidelberg **(2003)**.
[2] D. Boneh, C. Gentry, H. Shacham, B. Lynn, Aggregate and verifiably encrypted signatures from bilinear maps. In: EUROCRPYT'03, LNCS, vol. 2656, pp. 416-432. Springer, Heidelberg **(2003)**.
[3] X. Cheng, J. Liu, X. Wang, Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In: ICCSA'05, LNCS, vol. 3483, pp. 1046-1054. Springer, Heidelberg **(2005)**.
[4] C. Gentry, Z. Ramzan, Identity-based aggregate signature. In: PKC'06, LNCS, vol. 3958, pp. 257-273. Springer, Heidelberg **(2006)**.
[5] Z. Gong, Y. Long, X. Hong, K. Chen, Two certificateless aggregate signatures from bilinear maps. In: Eighth ACIS international conference on software engineering，artificial intelligence， networking， and parallel/distributed computing'07, pp. 188-193
[6] D. He, M. Tian, A note on 'An efficient certificateless aggregate signature with constant pairing computations'. Cryptology eprint Archive, Available online: *http://eprint. iacr.org/2012/445*.
[7] S.T. Kent, C. Lynn, J. Mikkelson, K. Seo, Secure border gateway protocol (S-BGP)-Real world performance and deployment issues. NDSS, Internet Society **(2000)**.
[8] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, B. Waters, Sequential aggregate signatures and multisignatures without random oracles. In: EUROCRPYT'06, LNCS, vol. 4004, pp. 465-485. Springer, Heidelberg **(2006)**.
[9] G. Neven, Efficient sequential aggregate signed date. In: EUROCRPYT'08, LNCS, vol. 4965, pp. 52-69. Springer, Heidelberg **(2008)**.
[10] [10] M. Ruckert, D. Schrode, Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In: ISA'09, LNCS, vol. 5576, pp. 750-759. Springer, Heidelberg **(2009)**.
[11] R. Sakai, K. Ohgishi, and M. Kasahara, Cryptosystems based on pairing. 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, **(2000)**, pp. 26-28.
[12] Z. Shao, Enhanced aggregate signature from pairings. In: CISC'05, LNCS, vol. 3822, pp. 140-149. Springer, Heidelberg **(2005)**.
[13] K. Shim, An ID-based aggregate signatue scheme with constant pairing computations. The Journal of System and Software, vol. 83 **(2010)**, pp. 1873-1880.
[14] H. Xiong, Z. Guan, Z. Chen, F. Li, An efficient certificateless aggregate signature with constant pairing computations. Information Sciences, vol. 10, **(2013)**, pp. 225-235.
[15] N. Yanai, R. Tso, M. Mambo, E. Okamoto, Certificateless ordered sequential aggregate signature scheme. In: Third international conference on intelligent networking and collaborative systems **(2011)**, pp. 662-667
[16] L. Zhang, F. Zhang, A new certificateless aggregate signature scheme. Computer Communication vol. 32 **(2009)**, pp. 1079-1085.
[17] B. Kang, On the security of some aggregate signature schemes. Journal of Applied Mathematics, Vol. 2012, Article ID 416137.
[18] B. Kang, ID-based aggregate signature scheme with constant pairing computations: attack and new construction. Journal of Computational Information Systems, no. 16 **(2012)**, pp. 6611-6618
[19] A. Yavuz, P. Ning, Hash-based sequential aggregate and forward secure signature for unattended wireless sensor networks, 2009 6th Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous 2009, 2009, 2009 6th Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous **(2009)**.
[20] M. Abdalla, S. Miner, C. Namprempre, Forward-secure threshold signature schemes, in: Topics in

Cryptology – CT-RSA 2001, LNCS, vol. 2020, Springer, Heidelberg, **(2001**), pp. 441–456.
[21] Q. Zhang, Z. Chen, A weighted kernel possibilistic means algorithm based on cloud computing for clustering big data. International Journal of Communication Systems, vol. 27, no. 9, (**2014),** pp. 1378-1391.

Authors

Baoyuan Kang, He received his M.S. in algebra from the shanxi University, and ph.D. in cryptography from Xidian University, People's Republic of China in 1993 and 1999, respectively. From 1993 to 1999, he taught mathematics in Northwestern Polytechnic University. Since 1999 he has taught mathematics and computer science in Central South University. Now he is a professor at Tianjin Polytechnic University. His current research interests are cryptography and information security.



Danhui Xu, She received her B.S. in Computer Science from the Tianjin Polytechnic University University, China in 2013. Now he is a postgraduate student at Tianjin Polytechnic University. His current research interests are cryptography and information security.

