Research Article

Reattack of a Certificateless Aggregate Signature Scheme with Constant Pairing Computations

Hang Tu,¹ Debiao He,² and Baojun Huang²

¹ School of Computer, Wuhan University, Wuhan 430072, China

² School of Mathematics and Statistics, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Debiao He; hedebiao@163.com

Received 20 September 2013; Accepted 9 January 2014; Published 13 March 2014

Academic Editors: T. M. Deserno and W. Zuo

Copyright © 2014 Hang Tu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A new attack against a novel certificateless aggregate signature scheme with constant pairing computations is presented. To enhance security, a new certificateless signature scheme is proposed first. Then a new certificateless aggregate signature scheme with constant pairing computations based on the new certificateless signature scheme is presented. Security analysis shows that the proposed certificateless aggregate signature scheme is provably secured in the random oracle.

1. Introduction

The concept of aggregate signature (AS) scheme was first introduced by Boneh et al. [1] in 2003. Such a scheme greatly reduces the computational and communication overhead since it could aggregate n signatures on n distinct messages from n distinct users into a single signature and check the correctness through a verification operation. The AS scheme is very useful in real-world applications. For example, in the scenario of the secure Border Gateway Protocol (BGP) [2], each router successively signs its own segment of a path in the network and then forwards the collection of signatures associated with the path to the next router. The AS scheme can be used to compress these signatures into a single one and hence reduce the overheads of both bandwidth and computation required in the original secure BGP. Similarly, in the scenario of the Vehicular Ad Hoc Networks (VANETs) [3], each vehicle or roadside unit (RSU) has to verify around 500–2000 messages per second. The AS scheme can be used to compress these messages into a single one and check the correctness through a verification operation. To satisfy different applications, many AS schemes based on the traditional public key cryptography (TPKC) and identity-based public key cryptography (ID-based PKC) have been proposed.

It is well known that a certificate, generated by a trusted third party, is needed to bind a user's identity and its public key in the TPKC. However, the management of these certificates becomes more and more difficult with the growth of users' number. To solve the problem, Shamir [4] introduced the concept of the ID-based PKC. In such cryptography, the user's identity, such as name, email address, and telephone number, is his public key and his private key is generated by the key generation centre (KGC) using his identity. However, the key escrow problem exists in the ID-based PKC since the KGC know the user's private key. In 2003, Al-Riyami and Paterson [5] developed the concept of the CLPKC to solve the key escrow problem in the ID-PKC. In CLPKC, the KGC only generates a partial private key for a user and the full private key of the user is a combination of his partial private key and some secret value chosen by the user himself. Recently, CLPKC attracted much attention and many certificateless encryption (CLE) schemes [6–8], certificateless key agreement (CLKA) schemes [9–11], certificateless signcryption (CLSC) schemes [12, 13], and certificateless signature (CLS) schemes [14–16] were proposed.

To satisfy the applications in certificateless environment, the certificateless aggregate signature (CLAS) scheme has attracted much attention. Several CLAS schemes [17–23] have been proposed by different researchers. However, most of

these schemes [17–20, 23] have computational complexity for pairing computations that grows linearly with the number of signers, which deviates from the main goals of aggregate signatures. Besides, both of the schemes [20, 22] of Zhang et al. require certain synchronization; that is, all signers must share the same synchronized clocks to generate aggregate signature. It is easy to say that it is difficult to achieve synchronization in many communication scenarios. Shim [24] pointed out that L. Zhang and F. Zhang's scheme [20] is vulnerable to the coalition attack. Xiong et al. [25] found that Hu et al.'s scheme [21] cannot provide unforgeability. Very recently, Xiong et al. [25] proposed a certificateless signature (CLS) scheme and constructed a new CLAS scheme using that CLS scheme. Compared to previous CLAS schemes, Xiong et al.'s CLAS scheme is very efficient in computation, and the verification procedure needs only a very small constant number of pairing operations, independent of the number of aggregated signatures. Besides, their scheme does not require synchronization for aggregating randomness. Unfortunately, He et al. [26] pointed out that Xiong et al.'s CLAS scheme is insecure against a Type II adversary by giving concrete attack. However, He et al. did not give countermeasure to enhance security. In this paper, we propose a new attack against Xiong et al.'s CLAS scheme; that is, a Type II adversary could forge legal signature for any message. To improve security, we also propose an improved CLAS scheme.

The organization of the paper is sketched as follows. Section 2 gives some preliminaries of the paper. Sections 3 and 4 review and analyze Xiong et al.'s scheme. Section 5 gives our improved scheme. Sections 6 and 7 discuss security and performance analysis of our scheme. At last, we give some conclusion in Section 8.

2. Preliminaries

2.1. Bilinear Pairing. Let G_1 be a cyclic additive group of prime order q and G_2 a cyclic multiplicative group of the same order q . We let P denote the generator of G_1 . A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_2$ which satisfies the following properties.

(1) Bilinearity

$$e(aQ, bR) = e(Q, R)^{ab}, \quad (1)$$

where $Q, R \in G_1$, $a, b \in \mathbb{Z}_q^*$.

(2) Nondegeneracy

$$e(P, P) \neq 1_{G_2}. \quad (2)$$

(3) Computability: there is an efficient algorithm to compute $e(Q, R)$ for all $Q, R \in G_1$.

The Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such admissible pairings. The following problems are assumed to be intractable within polynomial time.

Computational Diffie-Hellman (CDH) Problem. Given $aP, bP \in G_1$, the task of CDH problem is to compute abP , where P denotes the generator of G_1 .

2.2. Formal Model of CLS and CLAS. In this subsection, we will review the definition and security notions specified in [25], only with slight notational differences. There are two kinds of adversaries in the CLS scheme and the CLAS scheme, that is, the Type I adversary $\mathcal{A}1$ and the Type II adversary $\mathcal{A}2$. The adversary $\mathcal{A}1$ is not able to access the master key but he could replace public keys at his will. The adversary $\mathcal{A}2$ represents a malicious KGC who generates partial private key of users. $\mathcal{A}2$ could have access to the master key of KGC, but he is not able to replace public keys. The following are five oracles which can be accessed by the adversaries.

- (i) *CreateUser*: Given an identity ID_i , if ID_i has already been created, nothing is to be carried out. Otherwise, the oracle generates the partial private key psk_{ID_i} , the secret key usk_{ID_i} , and the public key upk_{ID_i} . It then stores the tuple $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ into a list L . In both cases, psk_{ID_i} is returned.
- (ii) *RevealPartialKey*: On input of an identity ID_i , the oracle searches L for a corresponding entry to ID_i . If it is not found, $\perp$ is returned; otherwise, the corresponding psk_{ID_i} is returned.
- (iii) *RevealSecertKey*: On input of an identity ID_i , the oracle searches L for a corresponding entry to ID_i . If it is not found, $\perp$ is returned; otherwise, the corresponding usk_{ID_i} is returned.
- (iv) *ReplaceKey*: On input of an identity ID_i and a user public/secret key pair $(upk'_{ID_i}, usk'_{ID_i})$, the oracle searches L for the entry of ID_i . If it is not found, nothing will be carried out. Otherwise, the oracle updates $(ID_i, psk_{ID_i}, upk_{ID_i}, usk_{ID_i})$ to $(ID_i, psk_{ID_i}, upk'_{ID_i}, usk'_{ID_i})$.
- (v) *Sign*: On input of a message m_i for ID_i , the signing oracle proceeds in one of the three cases below.
 - (a) A valid signature σ_i returned if ID_i has been created but the user public/secret key pair (usk_{ID_i}, upk_{ID_i}) has not been replaced.
 - (b) If ID_i has not been created, a symbol $\perp$ is returned.
 - (c) If the user public/secret key pair of ID_i has been replaced with say $(upk'_{ID_i}, usk'_{ID_i})$, then the oracle returns the result of $Sign(usk'_{ID_i}, psk_{ID_i}, m_i)$.

The security for a CLS scheme and a CLAS scheme is defined via the following two games separately.

Game 1. The first game is performed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}1, \mathcal{A}2\}$ for a CLS scheme as follows.

- (i) $\mathcal{C}$ executes *MasterKeyGen* to get master private/public key pair (mpk, msk) .
- (ii) $\mathcal{A}$ can adaptively issue the *CreateUser*, *RevealPartialKey*, *RevealSecertKey*, *ReplaceKey*, and *Sign* queries to $\mathcal{C}$.

- (iii) $\mathcal{A}$ is to output a message m_i^* and a signature σ_i^* corresponding to a target identity ID_i^* and a public key $upk_{ID_i^*}$.

We say that $\mathcal{A}$ wins Game 1, if and only if the following three conditions hold.

- (1) σ_i^* is a valid signature on messages m_i^* under identities ID_i^* and the corresponding public key $upk_{ID_i^*}$.
- (2) If $\mathcal{A}$ is $\mathcal{A}1$, the identity ID_i^* has not submitted to *RevealPartialKey* queries to get the partial private key $psk_{ID_i^*}$. If $\mathcal{A}$ is $\mathcal{A}2$, ID_i^* has not submitted to *RevealSecertKey* queries or *ReplaceKey* queries to get the secret key $usk_{ID_i^*}$.
- (3) The oracle *Sign* has never been queried with (ID_i^*, m_i^*) .

Definition 1. A CLS scheme is said to be secure if there is no probabilistic polynomial-time adversary $\mathcal{A} \in \{\mathcal{A}1, \mathcal{A}2\}$, which wins Game 1 with nonnegligible advantage.

Game 2. The second game is performed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}1, \mathcal{A}2\}$ for a CLAS scheme as follows.

- (i) $\mathcal{C}$ executes *MasterKeyGen* to get master private/public key pair (mpk, msk) .
- (ii) $\mathcal{A}$ can adaptively issue the *CreateUser*, *RevealPartialKey*, *RevealSecertKey*, *ReplaceKey*, and *Sign* queries to $\mathcal{C}$.
- (iii) $\mathcal{A}$ outputs a set of n users whose identities are from the set $L_{ID}^* = \{ID_1^*, \dots, ID_n^*\}$ and corresponding public keys from the set $L_{upk}^* = \{upk_1^*, \dots, upk_n^*\}$, n messages $L_m^* = \{m_1^*, \dots, m_n^*\}$, and an aggregate signature σ^* .

We say that $\mathcal{A}$ wins Game 2, if and only if the following three conditions hold.

- (1) σ_i^* is a valid aggregate signature on messages $\{m_1^*, \dots, m_n^*\}$ under identities $\{ID_1^*, \dots, ID_n^*\}$ and the corresponding public key $\{upk_1^*, \dots, upk_n^*\}$.
- (2) If $\mathcal{A}$ is $\mathcal{A}1$, at least one of the identities ID_i^* has not submitted to *RevealPartialKey* queries to get the partial private key $psk_{ID_i^*}$. If $\mathcal{A}$ is $\mathcal{A}2$, at least one of ID_i^* has not been submitted to *RevealSecertKey* queries or *ReplaceKey* queries to get the secret key $usk_{ID_i^*}$.
- (3) The oracle *Sign* has never been queried with (ID_i^*, m_i^*) .

Definition 2. A CLAS scheme is said to be secure if there is no probabilistic polynomial-time adversary $\mathcal{A} \in \{\mathcal{A}1, \mathcal{A}2\}$, which wins Game 2 with nonnegligible advantage.

3. Review of Xiong et al.'s Schemes

3.1. Xiong et al.'s CLS Scheme. In this subsection, we will briefly review Xiong et al.'s CLS scheme. Their CLS scheme consists of five algorithms: *MasterKeyGen*, *PartialKeyGen*, *UserKeyGen*, *Sign*, and *Verify*. The detail of these algorithms is described as follows.

MasterKeyGen. Given a security parameter k , KGC runs the algorithm as follows.

- (1) Generate a cyclic additive group G_1 and a cyclic multiplicative group G_2 with prime order q .
- (2) Generate two generators P, Q of G_1 and an admissible pairing $e : G_1 \times G_1 \rightarrow G_2$.
- (3) Generate a random number $s \in Z_q^*$ and compute $P_{pub} = sP$.
- (4) Choose cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow G_1$ and $H_2 : \{0, 1\}^* \rightarrow Z_q^*$.
- (5) KGC publishes the system parameters $\{q, G_1, G_2, e, P, Q, P_{pub}, H_1, H_2\}$ and key the master key s secretly.

PartialKeyGen. Given a user's identity ID_i , KGC computes the user's partial private key $psk_{ID_i} = sQ_{ID_i}$ and transmits it to the user secretly, where $Q_{ID_i} = H_1(ID_i)$.

UserKeyGen. The user with identity ID_i selects a random number $x_{ID_i} \in Z_q^*$ as his secret key usk_{ID_i} and computes his public key as $upk_{ID_i} = usk_{ID_i} \cdot P$.

Sign. Given a message m_i , the partial private key psk_{ID_i} , the secret key usk_{ID_i} , the user with identity ID_i , and the corresponding public key upk_{ID_i} , perform the following steps to generate a signature.

- (1) Generate a random number $r_i \in Z_q^*$ and compute $U_i = r_i P$.
- (2) Compute $h_i = H_2(m_i, ID_i, upk_{ID_i}, U_i)$, $V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot x_{ID_i} \cdot Q$.
- (3) Output (U_i, V_i) as the signature on m_i .

Verify. Given a signature (U_i, V_i) of message m_i on identity ID_i and corresponding public key upk_{ID_i} :

- (1) Compute $Q_{ID_i} = H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, upk_{ID_i}, U_i)$.
- (2) Verify $e(V_i, P) = e(h_i \cdot U_i + Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, Q)$ holds or not. If it holds, accept the signature.

3.2. Xiong et al.'s CLAS Scheme. In this subsection, we will briefly review Xiong et al.'s CLAS scheme. Their CLAS scheme consists of six algorithms: *MasterKeyGen*, *PartialKeyGen*, *UserKeyGen*, *Sign*, *Aggregate*, and *AggregateVerify*. The first four algorithms are the same as those in their CLS scheme. The detail of the other two algorithms is described as follows.

Aggregate. For an aggregating set of n users $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ with identities $\{ID_1, \dots, ID_n\}$, corresponding public keys

$\{upk_1, \dots, upk_n\}$, and message-signature pairs $\{(m_1, \sigma_1 = (U_1, V_1)), \dots, (m_n, \sigma_n = (U_n, V_n))\}$ from $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$, respectively, the aggregate signature generator computes $V = \sum_{i=1}^n V_i$ and outputs $\sigma = (U_1, \dots, U_n, V)$ as an aggregate signature.

AggregateVerify. To verify an aggregate signature $\sigma = (U_1, \dots, U_n, V)$ signed by n users $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ with identities $\{ID_1, \dots, ID_n\}$ and the corresponding public keys $\{upk_1, \dots, upk_n\}$ on messages $\{m_1, \dots, m_n\}$, the verifier performs the following steps.

- (1) Compute $Q_{ID_i} = H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, upk_{ID_i}, U_i)$ for $i = 1, \dots, n$.
- (2) Verify $e(V, P) = e(\sum_{i=1}^n (h_i \cdot U_i + Q_{ID_i}), P_{pub}) e(\sum_{i=1}^n h_i \cdot upk_{ID_i}, Q)$ holds or not. If it holds, accept the signature.

4. Cryptanalysis of Xiong et al.'s Scheme

Shim [24] claimed that both of their schemes are provably secure against two types of adversary in the random oracle model. However, in this section, we will disprove their claims by giving two concrete attacks.

4.1. Attack against Xiong et al.'s CLS Scheme. Shim [24] claimed their CLS scheme is semantically secure against Type II adversary. Unfortunately, it is not true, since there is a polynomial time Type II adversary $\mathcal{A}_2$ who can always win Game 1 through either of the following two attacks.

In Xiong et al.'s CLS scheme, the system parameters $\{q, G_1, G_2, e, P, Q, P_{pub}, H_1, H_2\}$ are generated by KGC. Let a Type II adversary $\mathcal{A}_2$ be a malicious KGC. Then $\mathcal{A}_2$ could choose a random t and compute $Q = tP$, when he generates the system parameters. After that, $\mathcal{A}_2$ could forge a legal signature of any message.

- (1) The Type II adversary $\mathcal{A}_2$ has the master key s . Then, he could compute a user $\mathcal{U}_i$'s partial private key $psk_{ID_i} = sQ_{ID_i}$, where $Q_{ID_i} = H_1(ID_i)$.
- (2) For any message m_i , $\mathcal{A}_2$ generates a random number $r_i \in \mathbb{Z}_q^*$ and computes $U_i = r_i P$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, U_i)$ and $V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}$.
- (3) $\mathcal{A}_2$ outputs (U_i, V_i) as the signature on m_i .

Since $Q = tP$, $U_i = r_i P$, and $V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}$, we could have

$$\begin{aligned} e(V_i, P) &= e(V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}, P) \\ &= e(psk_{ID_i} + h_i \cdot r_i \cdot P_{pub}, P) e(h_i \cdot t \cdot upk_{ID_i}, P) \\ &= e(sQ_{ID_i} + h_i \cdot r_i \cdot sP, P) e(h_i \cdot upk_{ID_i}, t \cdot P) \\ &= e(h_i \cdot U_i + Q_{ID_i}, sP) e(h_i \cdot upk_{ID_i}, Q) \\ &= e(h_i \cdot U_i + Q_{ID_i}, P_{pub}) e(h_i \cdot upk_{ID_i}, Q). \end{aligned} \quad (3)$$

Then, we know that (U_i, V_i) is a legal signature on m_i . Besides, ID_i has not been submitted to *RevealSecertKey* queries or *ReplaceKey* queries to get the secret key usk_{ID_i} and the oracle *Sign* has never been queried with (ID_i, m_i) . So the Type II adversary $\mathcal{A}_2$ wins Game 1.

Therefore, Xiong et al.'s CLS scheme is not secure against attacks of the Type II adversary.

4.2. Attack against Xiong et al.'s CLAS Scheme. Shim [24] claimed their CLAS scheme is semantically secure against Type II adversary. Unfortunately, it is not true, since there exists a polynomial time Type II adversary $\mathcal{A}_2$, who can always win Game 2 through either of the following two attacks.

In Xiong et al.'s CLAS scheme, the system parameters $\{q, G_1, G_2, e, P, Q, P_{pub}, H_1, H_2\}$ are generated by KGC. Let a Type II adversary $\mathcal{A}_2$ be a malicious KGC. Then $\mathcal{A}_2$ could choose a random t and compute $Q = tP$, when he generates the system parameters. After that, $\mathcal{A}_2$ could forge an aggregate signature.

Let $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ be an aggregating set of n users with identities $\{ID_1, \dots, ID_n\}$ and the corresponding public keys $\{upk_1, \dots, upk_n\}$.

- (1) For $i = 1, 2, \dots, n$, $\mathcal{A}_2$ does the following five substeps to generate a legal signature (U_i, V_i) on a message m_i .
 - (i) The Type II adversary $\mathcal{A}_2$ has the master key s . Then, he could compute a user $\mathcal{U}_i$'s partial private key $psk_{ID_i} = sQ_{ID_i}$, where $Q_{ID_i} = H_1(ID_i)$.
 - (ii) For any message m_i , $\mathcal{A}_2$ generates a random number $r_i \in \mathbb{Z}_q^*$ and computes $U_i = r_i P$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, U_i)$, and $V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}$.
 - (iii) $\mathcal{A}_2$ outputs (U_i, V_i) as the signature on m_i .

- (2) $\mathcal{A}_2$ computes $V = \sum_{i=1}^n V_i$.

- (3) $\mathcal{A}_2$ outputs $\sigma = (U_1, \dots, U_n, V)$ as an aggregate signature.

From the analysis in the above subsection, we know that (U_i, V_i) satisfies the equation $e(V_i, P) = e(h_i \cdot U_i + Q_{ID_i}, sP_{pub}) e(h_i \cdot upk_{ID_i}, Q)$, where $Q = tP$ and $V_i = psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}$. Then we could have that

$$\begin{aligned} e(V, P) &= e\left(\sum_{i=1}^n V_i, P\right) \\ &= e\left(\sum_{i=1}^n (psk_{ID_i} + h_i \cdot r_i \cdot P_{pub} + h_i \cdot t \cdot upk_{ID_i}), P\right) \\ &= e\left(\sum_{i=1}^n (psk_{ID_i} + h_i \cdot r_i \cdot P_{pub}), P\right) \\ &\quad \times e\left(\sum_{i=1}^n h_i \cdot t \cdot upk_{ID_i}, P\right) \end{aligned}$$

$$= e \left(\sum_{i=1}^n (h_i \cdot U_i + Q_{ID_i}), P_{pub} \right) \times e \left(\sum_{i=1}^n h_i \cdot upk_{ID_i}, Q \right). \quad (4)$$

Thus, we know that $\sigma = (U_1, \dots, U_n, V)$ is a legal aggregate signature on messages $\{m_1, \dots, m_n\}$. Besides, for any $i \in \{1, \dots, n\}$, ID_i has not been submitted to *RevealSecertKey* queries or *ReplaceKey* queries to get the secret key usk_{ID_i} and the oracle *Sign* has never been queried with (ID_i, m_i) . So the Type II adversary $\mathcal{A}_2$ wins Game 2.

Therefore, Xiong et al.'s CLAS scheme is not secure against attacks of the Type II adversary.

5. Our CLS Scheme and CLAS Scheme

5.1. Our CLS Scheme. Like Xiong et al.'s CLS scheme does, our CLS scheme also consists of five algorithms: *MasterKeyGen*, *PartialKeyGen*, *UserKeyGen*, *Sign*, and *Verify*. The detail of these algorithms is described as follows.

MasterKeyGen. Given a security parameter k , KGC runs the algorithm as follows.

- (1) Generate a cyclic additive group G_1 and a cyclic multiplicative group G_2 with prime order q .
- (2) Generate a generator P of G_1 and an admissible pairing $e : G_1 \times G_1 \rightarrow G_2$.
- (3) Generate a random number $s \in \mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
- (4) Choose cryptographic hash functions $H_1, H_2, H_3 : \{0, 1\}^* \rightarrow G_1$ and $H_4, H_5 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$.
- (5) KGC publishes the system parameters $params = \{q, G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$ and key the master key s secretly.

PartialKeyGen. Given a user's identity ID_i , KGC computes the user's partial private key $psk_{ID_i} = sQ_{ID_i}$ and transmits it to the user secretly, where $Q_{ID_i} = H_1(ID_i)$.

UserKeyGen. The user with identity ID_i selects a random number $x_{ID_i} \in \mathbb{Z}_q^*$ as his secret key usk_{ID_i} and computes his public key as $upk_{ID_i} = usk_{ID_i} \cdot P$.

Sign. Given a message m_i , the partial private key psk_{ID_i} , the secret key usk_{ID_i} , the user with identity ID_i , and the corresponding public key upk_{ID_i} , perform the following steps to generate a signature.

- (1) Generate a random number $r_i \in \mathbb{Z}_q^*$ and compute $U_i = r_i P$.
- (2) Compute $W = H_2(params)$, $T = H_3(params)$, $h_i = H_4(m_i, ID_i, upk_{ID_i}, U_i)$, and $k_i = H_5(m_i, ID_i, upk_{ID_i}, U_i)$.
- (3) Compute, $V_i = psk_{ID_i} + h_i \cdot x_{ID_i} \cdot W + k_i \cdot r_i \cdot T$.
- (4) Output (U_i, V_i) as the signature on m_i .

Verify. Given a signature (U_i, V_i) of message m_i on identity ID_i and corresponding public key upk_{ID_i} , consider the following.

- (1) Compute $Q_{ID_i} = H_1(ID_i)$, $W = H_2(params)$, $T = H_3(params)$, $h_i = H_4(m_i, ID_i, upk_{ID_i}, U_i)$, and $k_i = H_5(m_i, ID_i, upk_{ID_i}, U_i)$.
- (2) Verify $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, W) e(k_i U_i, T)$ holds or not. If it holds, accept the signature.

5.2. Our CLAS Scheme. Like Xiong et al.'s CLAS scheme does, our CLAS scheme also consists of six algorithms: *MasterKeyGen*, *PartialKeyGen*, *UserKeyGen*, *Sign*, *Aggregate*, and *AggregateVerify*. The first four algorithms are the same as those in our CLS scheme. The detail of other two algorithms is described as follows.

Aggregate. For an aggregating set of n users $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ with identities $\{ID_1, \dots, ID_n\}$, corresponding public keys $\{upk_1, \dots, upk_n\}$, and message-signature pairs $\{(m_1, \sigma_1 = (U_1, V_1)), \dots, (m_n, \sigma_n = (U_n, V_n))\}$ from $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$, respectively, the aggregate signature generator computes $V = \sum_{i=1}^n V_i$ and outputs $\sigma = (U_1, \dots, U_n, V)$ as an aggregate signature.

AggregateVerify. To verify an aggregate signature $\sigma = (U_1, \dots, U_n, V)$ signed by n users $\{\mathcal{U}_1, \dots, \mathcal{U}_n\}$ with identities $\{ID_1, \dots, ID_n\}$ and the corresponding public keys $\{upk_1, \dots, upk_n\}$ on messages $\{m_1, \dots, m_n\}$, the verifier performs the following steps.

- (1) Compute $Q_{ID_i} = H_1(ID_i)$, $W = H_2(params)$, $T = H_3(params)$, $h_i = H_4(m_i, ID_i, upk_{ID_i}, U_i)$, and $k_i = H_5(m_i, ID_i, upk_{ID_i}, U_i)$ for $i = 1, \dots, n$.
- (2) Verify $e(V, P) = e(\sum_{i=1}^n Q_{ID_i}, P_{pub})e(\sum_{i=1}^n h_i \cdot upk_{ID_i}, Q)e(\sum_{i=1}^n k_i \cdot U_i, T)$ holds or not. If it holds, accept the signature.

6. Security Analysis

6.1. Security Analysis of Our CLS Scheme. In this section, we analyze the security of our CLS scheme. The following lemmas and theorem are proposed.

Lemma 3. *If Type I adversary $\mathcal{A}_1$ wins Game 1 with nonnegligible probability ϵ , then one could construct an algorithm to solve the CDH problem in G_1 with nonnegligible probability.*

Proof. Given an instance $(X = xP, Y = yP)$ of the CDH problem in G_1 , we will construct an algorithm $\mathcal{C}$ to compute xyP , where $x, y \in \mathbb{Z}_q^*$ and they are unknown to $\mathcal{C}$. At first, $\mathcal{C}$ picks an identity ID_1 at random as the challenged identity in this game, sets the master public key $P_{pub} = X$, selects the system parameters $params = \{G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$, and returns the parameters to $\mathcal{A}_1$. Then $\mathcal{C}$ answers $\mathcal{A}_1$'s query as follows.

- (i) H_1 query: $\mathcal{C}$ maintains a list L_{H_1} of form $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$. When $\mathcal{A}_1$ makes this query on ID_i , $\mathcal{C}$ does as follows.

- (a) If the list L_{H_1} contains a tuple $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$, $\mathcal{E}$ returns Q_{ID_i} to $\mathcal{A}$ 1.
 - (b) Otherwise, if $ID_i = ID_I$, $\mathcal{E}$ picks a random $a_{ID_i} \in Z_n^*$, computes $Q_{ID_i} = a_{ID_i}Y$, adds $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ to L_{H_1} , and returns Q_{ID_i} to $\mathcal{A}$ 1.
 - (c) Otherwise ($ID_i \neq ID_I$), $\mathcal{E}$ picks a random $a_{ID_i} \in Z_n^*$, computes $Q_{ID_i} = a_{ID_i}P$, adds $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ to L_{H_1} , and returns Q_{ID_i} to $\mathcal{A}$ 1.
- (ii) H_2 query: $\mathcal{E}$ maintains a list L_{H_2} of form $\langle m_i, b_i, W_i \rangle$. When $\mathcal{A}$ 1 makes this query on m_i , $\mathcal{E}$ does as follows.
- (a) If the list L_{H_2} contains a tuple $\langle m_i, b_i, W_i \rangle$, $\mathcal{E}$ returns W_i to $\mathcal{A}$ 1.
 - (b) Otherwise, $\mathcal{E}$ picks a random $b_i \in Z_n^*$, computes $W_i = b_iP$, adds $\langle m_i, b_i, W_i \rangle$ to L_{H_2} , and returns W_i to $\mathcal{A}$ 1.
- (iii) H_3 query: $\mathcal{E}$ maintains a list L_{H_3} of form $\langle m_i, c_i, T_i \rangle$. When $\mathcal{A}$ 1 makes this query on m_i , $\mathcal{E}$ does as follows.
- (a) If the list L_{H_3} contains a tuple $\langle m_i, c_i, T_i \rangle$, $\mathcal{E}$ returns T_i to $\mathcal{A}$ 1.
 - (b) Otherwise, $\mathcal{E}$ picks a random $c_i \in Z_n^*$, computes $T_i = c_iX$, adds $\langle m_i, c_i, T_i \rangle$ to L_{H_3} , and returns T_i to $\mathcal{A}$ 1.
- (iv) H_4 query: $\mathcal{E}$ maintains a list L_{H_4} of form $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$. When $\mathcal{A}$ 1 makes this query on $(m_i, ID_i, upk_{ID_i}, U_i)$, $\mathcal{E}$ does as follows.
- (a) If the list L_{H_4} contains a tuple $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$, $\mathcal{E}$ returns h_i to $\mathcal{A}$ 1.
 - (b) Otherwise, $\mathcal{E}$ picks a random $h_i \in Z_n^*$, returns h_i to $\mathcal{A}$ 1, and adds $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ to L_{H_4} .
- (v) H_5 query: $\mathcal{E}$ maintains a list L_{H_5} of form $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$. When $\mathcal{A}$ 1 makes this query on $(m_i, ID_i, upk_{ID_i}, U_i)$, $\mathcal{E}$ does as follows.
- (a) If the list L_{H_5} contains a tuple $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$, $\mathcal{E}$ returns k_i to $\mathcal{A}$ 1.
 - (b) Otherwise, $\mathcal{E}$ picks a random $k_i \in Z_n^*$, returns k_i to $\mathcal{A}$ 1, and adds $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ to L_{H_5} .
- (vi) *CreateUser*: $\mathcal{E}$ maintains a list L_U of form $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$. When $\mathcal{A}$ 1 makes this query on ID_i , $\mathcal{E}$ does as follows.
- (a) If the list L_U contains a tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$, $\mathcal{E}$ returns upk_{ID_i} to $\mathcal{A}$ 1.
 - (b) Otherwise, if $ID_i = ID_I$, $\mathcal{E}$ gets $Q_{ID_i} = a_{ID_i}Y$ by making H_1 query with ID_I and sets $psk_{ID_i} \leftarrow \perp$. $\mathcal{E}$ selects a random number $x_{ID_i} \in Z_q^*$ as his secret key usk_{ID_i} and computes his public key as $upk_{ID_i} = usk_{ID_i} \cdot P$. At last, $\mathcal{E}$ adds $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ to L_U and returns upk_{ID_i} to $\mathcal{A}$ 1.
 - (c) Otherwise ($ID_i \neq ID_I$), $\mathcal{E}$ gets $Q_{ID_i} = a_{ID_i} \cdot P$ by making H_1 query with ID_i and sets $psk_{ID_i} = a_{ID_i} \cdot P_{pub}$. $\mathcal{E}$ selects a random number $x_{ID_i} \in Z_q^*$ as his secret key usk_{ID_i} and computes his public key as $upk_{ID_i} = usk_{ID_i} \cdot P$. At last, $\mathcal{E}$ adds $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ to L_U and returns upk_{ID_i} to $\mathcal{A}$ 1.
- (vii) *RevealPartialKey*: when $\mathcal{A}$ 1 makes this query on ID_i , $\mathcal{E}$ does as follows.
- (a) If $ID_i = ID_I$, $\mathcal{E}$ stops the simulation.
 - (b) Otherwise ($ID_i \neq ID_I$), $\mathcal{E}$ looks up the list L_U for the tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ and returns psk_{ID_i} to $\mathcal{A}$ 1.
- (viii) *RevealSecertKey*: when $\mathcal{A}$ 1 makes this query on ID_i , $\mathcal{E}$ looks up the list L_U for the tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ and returns usk_{ID_i} to $\mathcal{A}$ 1.
- (ix) *ReplaceKey*: when $\mathcal{A}$ 1 makes this query on (ID_i, upk'_{ID_i}) , $\mathcal{E}$ looks up the list L_U for the tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$. $\mathcal{E}$ sets $usk_{ID_i} \leftarrow \perp$ and replaces upk_{ID_i} with upk'_{ID_i} .
- (x) *Sign*: when $\mathcal{A}$ 1 makes this query on (m_i, ID_i) , $\mathcal{E}$ does as follows.
- (a) If $ID_i = ID_I$, $\mathcal{E}$ looks up lists L_{H_1} and L_U and for tuples $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ and $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ separately, where $Q_{ID_i} = a_{ID_i}Y$ and upk_{ID_i} may have been replaced by $\mathcal{A}$ 1. $\mathcal{E}$ makes H_2 query and H_3 query with $params$ and gets tuples $\langle params, b_i, W \rangle$ and $\langle params, c_i, T \rangle$, where $W = b_iP$ and $T = c_iX$. $\mathcal{E}$ generates three random numbers $h_i, k_i, r_i \in Z_q^*$, computes $U_i = k_i^{-1}(r_i \cdot P - a_{ID_i} \cdot c_i^{-1} \cdot Y)$, $V_i = r_i \cdot T + h_i \cdot b_i \cdot upk_{ID_i}$, and sets $H_4(m_i, ID_i, upk_{ID_i}, U_i) \leftarrow h_i$, $H_5(m_i, ID_i, upk_{ID_i}, U_i) \leftarrow k_i$. At last, $\mathcal{E}$ adds tuples $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ and $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$ to L_{H_4} and L_{H_5} separately and returns (U_i, V_i) to $\mathcal{A}$ 1. It is easy to say (U_i, V_i) satisfies the equation $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, W)e(k_i U_i, T)$.
 - (b) Otherwise ($ID_i \neq ID_I$), $\mathcal{E}$ looks up lists L_{H_1} and L_U and for tuples $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ and $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ separately, where $Q_{ID_i} = a_{ID_i}P$ and upk_{ID_i} may have been replaced by $(\mathcal{A}$ 1). $\mathcal{E}$ makes H_2 query and H_3 query with $params$ and gets tuples $\langle params, b_i, W \rangle$ and $\langle params, c_i, T \rangle$, where $W = b_iP$ and $T = c_iX$. $\mathcal{E}$ generates a random number $r_i \in Z_q^*$ and computes $U_i = r_iP$, $h_i = H_4(m_i, ID_i, upk_{ID_i}, U_i)$, $k_i = H_5(m_i, ID_i, upk_{ID_i}, U_i)$, and $V_i = psk_{ID_i} + h_i \cdot b_i \cdot upk_{ID_i} + k_i \cdot r_i \cdot T$. At last, $\mathcal{E}$ returns (U_i, V_i) to $\mathcal{A}$ 1. It is easy to say (U_i, V_i) satisfies the equation $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, W)e(k_i U_i, T)$.

Finally, $\mathcal{A}1$ outputs a tuple (ID_i, m_i, σ_i) as its forgery, where $\sigma_i = (U_i, V_i)$. If $ID_i \neq ID_I$, $\mathcal{C}$ stops the simulation. From the forgery lemma [27], if we have a replay of $\mathcal{C}$ with the same random tape but different choice of H_4 and H_5 , $\mathcal{A}1$ will output another three signatures. The following two equations hold since both of the two signatures are valid:

$$\begin{aligned} e(V_i, P) &= e(Q_{ID_i}, P_{pub}) e(h_i \cdot upk_{ID_i}, W) e(k_i U_i, T), \\ e(V'_i, P) &= e(Q_{ID_i}, P_{pub}) e(h'_i \cdot upk_{ID_i}, W) e(k'_i U_i, T). \end{aligned} \quad (5)$$

$\mathcal{C}$ looks up lists L_{H_1} and L_U and for tuples $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ and $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ separately, where $Q_{ID_i} = a_{ID_i} P$ and upk_{ID_i} may have been replaced by $\mathcal{A}1$. $\mathcal{C}$ makes H_2 query and H_3 query with $params$ and gets tuples $\langle params, b_i, W \rangle$ and $\langle params, c_i, T \rangle$, where $W = b_i P$ and $T = c_i X$. At last, $\mathcal{C}$ returns $(k_i(V'_i - b_i \cdot h'_i \cdot upk_{ID_i}) - k'_i(V_i - b_i \cdot h_i \cdot upk_{ID_i})) / a_{ID_i}(k_i - k'_i)$ as the solution of CDH problem. $\square$

Analysis. We show that $\mathcal{C}$ solves the given instance of the CDH problem with the probability η . To do so, we analyze the three events that result in $\mathcal{C}$'s success.

E_1 : $\mathcal{C}$ does not abort in all the queries of *RevealPartialKey*.

E_2 : $\mathcal{A}1$ can forge a legal signature $\sigma_i = (U_i, V_i)$.

E_3 : the outputted tuple (ID_i, m_i, σ_i) satisfies $ID_i = ID_I$.

From the simulation we know that $\Pr[E_1] \geq (1 - (1/q_{H_1}))^{q_{RPK}}$, $\Pr[E_2 \mid E_1] \geq \varepsilon$, $\Pr[E_3 \mid E_1 \wedge E_2] \geq (1/q_{H_1})$, where q_{H_1} and q_{RPK} denote the numbers of H_1 queries and *RevealPartialKey* queries separately. Then, the probability that $\mathcal{C}$ solves the CDH problem is

$$\begin{aligned} \eta &= \Pr[E_1 \wedge E_2 \wedge E_3] \\ &= \Pr[E_1] \Pr[E_2 \mid E_1] \Pr[E_3 \mid E_1 \wedge E_2] \\ &\geq \frac{1}{q_{H_1}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{EPPK}} \varepsilon. \end{aligned} \quad (6)$$

Then $\mathcal{C}$ could solve the CDH problem with a nonnegligible probability since ε is nonnegligible. This contradicts the hardness of the CDH problem. Therefore, our CLS scheme is existentially unforgeable against Type I adversary in random oracle model under the assumption that the CDH problem is hard.

Lemma 4. *If there is a Type I adversary $\mathcal{A}2$ wins Game 1 with nonnegligible probability ε . Then we could construct an algorithm $\mathcal{C}$ to solve the CDH problem in G_1 with nonnegligible probability.*

Proof. Given an instance $(X = xP, Y = yP)$ of the CDH problem in G_1 , we will construct an algorithm $\mathcal{C}$ to compute xyP , where $x, y \in Z_q^*$ and they are unknown to $\mathcal{C}$. At first, $\mathcal{C}$

picks an identity ID_I at random as the challenged identity in this game, generates a random number $s \in Z_q^*$ as the master key, sets the master public key $P_{pub} = sP$, selects the system parameters $params = \{G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$, and returns the master key and the parameters to $\mathcal{A}2$. Then $\mathcal{C}$ answers $\mathcal{A}2$'s query as follows.

(i) H_1 query: $\mathcal{C}$ maintains a list L_{H_1} of form $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$. When $\mathcal{A}1$ makes this query on ID_i , $\mathcal{C}$ does as follows.

(a) If the list L_{H_1} contains a tuple $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$, $\mathcal{C}$ returns Q_{ID_i} to $\mathcal{A}2$.

(b) Otherwise, $\mathcal{C}$ picks a random $a_{ID_i} \in Z_n^*$, computes $Q_{ID_i} = a_{ID_i} P$, adds $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ to L_{H_1} , and returns Q_{ID_i} to $\mathcal{A}2$.

(ii) H_2 query: $\mathcal{C}$ maintains a list L_{H_2} of form $\langle m_i, b_i, W_i \rangle$. When $\mathcal{A}2$ makes this query on m_i , $\mathcal{C}$ does as follows.

(a) If the list L_{H_2} contains a tuple $\langle m_i, b_i, W_i \rangle$, $\mathcal{C}$ returns W_i to $\mathcal{A}2$.

(b) Otherwise, $\mathcal{C}$ picks a random $b_i \in Z_n^*$, computes $W_i = b_i X$, adds $\langle m_i, b_i, W_i \rangle$ to L_{H_2} , and returns W_i to $\mathcal{A}2$.

(iii) H_3 query: $\mathcal{C}$ maintains a list L_{H_3} of form $\langle m_i, c_i, T_i \rangle$. When $\mathcal{A}2$ makes this query on m_i , $\mathcal{C}$ does as follows.

(a) If the list L_{H_3} contains a tuple $\langle m_i, c_i, T_i \rangle$, $\mathcal{C}$ returns T_i to $\mathcal{A}2$.

(b) Otherwise, $\mathcal{C}$ picks a random $c_i \in Z_n^*$, computes $T_i = c_i X$, adds $\langle m_i, c_i, T_i \rangle$ to L_{H_3} , and returns T_i to $\mathcal{A}2$.

(iv) H_4 query: $\mathcal{C}$ maintains a list L_{H_4} of form $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$. When $\mathcal{A}2$ makes this query on $(m_i, ID_i, upk_{ID_i}, U_i)$, $\mathcal{C}$ does as follows.

(a) If the list L_{H_4} contains a tuple $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$, $\mathcal{C}$ returns h_i to $\mathcal{A}2$.

(b) Otherwise, $\mathcal{C}$ picks a random $h_i \in Z_n^*$, returns h_i to $\mathcal{A}2$, and adds $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ to L_{H_4} .

(v) H_5 query: $\mathcal{C}$ maintains a list L_{H_5} of form $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$. When $\mathcal{A}2$ makes this query on $(m_i, ID_i, upk_{ID_i}, U_i)$, $\mathcal{C}$ does as follows.

(a) If the list L_{H_5} contains a tuple $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$, $\mathcal{C}$ returns k_i to $\mathcal{A}2$.

(b) Otherwise, $\mathcal{C}$ picks a random $k_i \in Z_n^*$, returns k_i to $\mathcal{A}2$, and adds $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ to L_{H_5} .

(vi) *CreateUser*: $\mathcal{C}$ maintains a list L_U of form $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$. When $\mathcal{A}2$ makes this query on ID_i , $\mathcal{C}$ does as follows.

(a) If the list L_U contains a tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$, $\mathcal{C}$ returns upk_{ID_i} to $\mathcal{A}2$.

- (b) Otherwise, if $ID_i = ID_I$, $\mathcal{C}$ gets $Q_{ID_i} = a_{ID_i}P$ by making H_1 query with ID_I and computes $psk_{ID_i} = sQ_{ID_i}$. $\mathcal{C}$ sets $usk_{ID_i} \leftarrow \perp$ and $upk_{ID_i} \leftarrow Y$. At last, $\mathcal{C}$ adds $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ to L_U and returns upk_{ID_i} to $\mathcal{A}2$.
- (c) Otherwise ($ID_i \neq ID_I$), $\mathcal{C}$ gets $Q_{ID_i} = a_{ID_i} \cdot P$ by making H_1 query with ID_i and computes $psk_{ID_i} = sQ_{ID_i}$. $\mathcal{C}$ selects a random number $x_{ID_i} \in Z_q^*$ as his secret key usk_{ID_i} and computes his public key as $upk_{ID_i} = usk_{ID_i} \cdot P$. At last, $\mathcal{C}$ adds $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ to L_U and returns upk_{ID_i} to $\mathcal{A}1$.
- (vii) *RevealPartialKey*: when $\mathcal{A}2$ makes this query on ID_i , $\mathcal{C}$ looks up the list L_U for the tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ and returns psk_{ID_i} to $\mathcal{A}2$.
- (viii) *RevealSecertKey*: when $\mathcal{A}2$ makes this query on ID_i , $\mathcal{C}$ does as follows.
- (a) If $ID_i = ID_I$, $\mathcal{C}$ stops the simulation.
- (b) Otherwise ($ID_i \neq ID_I$), $\mathcal{C}$ looks up the list L_U for the tuple $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ and returns usk_{ID_i} to $\mathcal{A}2$.
- (ix) *Sign*: when $\mathcal{A}1$ makes this query on (m_i, ID_i) , $\mathcal{C}$ does as follows.
- (a) If $ID_i = ID_I$, $\mathcal{C}$ looks up lists L_{H_1} and L_U and for tuples $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ and $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ separately, where $Q_{ID_i} = a_{ID_i}P$ and upk_{ID_i} may have been replaced by $\mathcal{A}2$. $\mathcal{C}$ makes H_2 query and H_3 query with $params$ and gets tuples $\langle params, b_i, W \rangle$ and $\langle params, c_i, T \rangle$, where $W = b_iX$ and $T = c_iX$. $\mathcal{C}$ generates three random numbers $h_i, k_i, r_i \in Z_q^*$, computes $U_i = k_i^{-1}(r_i \cdot P - b_i \cdot c_i^{-1} \cdot h_i \cdot Y)$, $V_i = r_i \cdot T + a_{ID_i} \cdot s \cdot upk_{ID_i}$, and sets $H_4(m_i, ID_i, upk_{ID_i}, U_i) \leftarrow h_i$, $H_5(m_i, ID_i, upk_{ID_i}, U_i) \leftarrow k_i$. At last, $\mathcal{C}$ adds tuples $\langle m_i, ID_i, upk_{ID_i}, U_i, h_i \rangle$ and $\langle m_i, ID_i, upk_{ID_i}, U_i, k_i \rangle$ to L_{H_4} and L_{H_5} separately and returns (U_i, V_i) to $\mathcal{A}2$. It is easy to say (U_i, V_i) satisfies the equation $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, W)e(k_i U_i, T)$.
- (b) Otherwise ($ID_i \neq ID_I$), $\mathcal{C}$ acts according to the description of the algorithm *Sign* since he knows both of usk_{ID_i} and psk_{ID_i} .

Finally, $\mathcal{A}2$ outputs a tuple (ID_i, m_i, σ_i) as its forgery, where $\sigma_i = (U_i, V_i)$. If $ID_i \neq ID_I$, $\mathcal{C}$ stops the simulation. From the forgery lemma [27], if we have a replay of $\mathcal{C}$ with the same random tape but different choice of H_4 and H_5 , $\mathcal{A}2$ will output another signatures. The following two equations hold since both of the two signatures are valid:

$$\begin{aligned} e(V_i, P) &= e(Q_{ID_i}, P_{pub})e(h_i \cdot upk_{ID_i}, W)e(k_i U_i, T), \\ e(V'_i, P) &= e(Q_{ID_i}, P_{pub})e(h'_i \cdot upk_{ID_i}, W)e(k'_i U_i, T). \end{aligned} \quad (7)$$

$\mathcal{C}$ looks up lists L_{H_1} and L_U and for tuples $\langle ID_i, a_{ID_i}, Q_{ID_i} \rangle$ and $\langle ID_i, psk_{ID_i}, Q_{ID_i}, usk_{ID_i}, upk_{ID_i} \rangle$ separately, where $Q_{ID_i} = a_{ID_i}P$. $\mathcal{C}$ makes H_2 query and H_3 query with $params$ and gets tuples $\langle params, b_i, W \rangle$ and $\langle params, c_i, T \rangle$, where $W = b_iX$ and $T = c_iX$. At last, $\mathcal{C}$ returns $(k_i(V'_i - s \cdot Q_{ID_i}) - k'_i(V_i - s \cdot Q_{ID_i}))/b_i(h'_i k_i - h_i k'_i)$ as the solution of CDH problem. $\square$

Analysis. We show that $\mathcal{C}$ solves the given instance of the CDH problem with the probability η . To do so, we analyze the three events that result in $\mathcal{C}$'s success.

E_1 : $\mathcal{C}$ does not abort in all the queries of *RevealSecertKey*.

E_2 : $\mathcal{A}1$ can forge a legal signature $\sigma_i = (U_i, V_i)$.

E_3 : the outputted tuple (ID_i, m_i, σ_i) satisfies $ID_i = ID_I$.

From the simulation we know that $\Pr[E_1] \geq (1 - (1/q_{H_1}))^{q_{RSK}}$, $\Pr[E_2 | E_1] \geq \varepsilon$, $\Pr[E_3 | E_1 \wedge E_2] \geq (1/q_{H_1})$, where q_{H_1} and q_{RSK} denote the numbers of H_1 queries and *RevealSecertKey* queries separately. Then, the probability that $\mathcal{C}$ solves the CDH problem is

$$\begin{aligned} \eta &= \Pr[E_1 \wedge E_2 \wedge E_3] \\ &= \Pr[E_1] \Pr[E_2 | E_1] \Pr[E_3 | E_1 \wedge E_2] \\ &\geq \frac{1}{q_{H_1}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{RSK}} \varepsilon. \end{aligned} \quad (8)$$

Then $\mathcal{C}$ could solve the CDH problem with a nonnegligible probability since ε is nonnegligible. This contradicts the hardness of the CDH problem. Therefore, our CLS scheme is existentially unforgeable against a Type II adversary in random oracle model under the assumption that the CDH problem is hard.

From the above two lemmas, we could get the following theorem.

Theorem 5. *The CLS scheme is secure against adaptively chosen warrant attacks and chosen message and identity attacks in the random oracle model if the CDH problem in G_1 is intractable.*

6.2. Security Analysis of Our CLAS Scheme. For the security of our CLAS scheme, we have the following theorem.

Theorem 6. *The CLAS scheme is secure against adaptively chosen warrant attacks and chosen message and identity attacks in the random oracle model if the CDH problem in G_1 is intractable.*

Proof. Suppose there is an adversary $\mathcal{A} \in \{\mathcal{A}1, \mathcal{A}2\}$ who could win Game 2 with nonnegligible probability. We could construct another adversary, who could win Game 1 with nonnegligible probability, through the method described in Theorem 2 of Xiong et al.'s work [25]. We have shown that no adversary could win Game 1 with nonnegligible probability

TABLE 1: Performance comparisons.

	Sign	Verify	Aggregate verify
Zhang et al.'s scheme [22]	$5T_S$	$5T_P + 2T_S$	$5T_P + 2nT_S$
Xiong et al.'s scheme [25]	$3T_S$	$3T_P + 2T_S$	$3T_P + 2nT_S$
Our scheme	$3T_S$	$4T_P + 2T_S$	$4T_P + 2nT_S$

in the above theorem. Therefore, our CLAS scheme is secure against adaptively chosen warrant attacks and chosen message and identity attacks in the random oracle model if the CDH problem in G_1 is intractable. $\square$

7. Performance Analysis

In this section, we compare our scheme with two latest CLAS schemes, that is, Zhang et al.'s scheme [22] and Xiong et al.'s scheme [25]. For convenience, some notations are defined as follows:

- (i) T_P : the time for executing a pairing operation;
- (ii) T_S : the time for executing a scalar multiplications in G_1 .

The comparisons are listed in Table 1. From the table, we know that Xiong et al.'s scheme and our scheme have better performance than Zhang et al.'s scheme. Xiong et al.'s scheme has better performance than our scheme. However, Xiong et al.'s scheme cannot withstand attacks of Type II adversary. It is well known that security is a top priority in network communications. It is acceptable to enhance security at the cost of increasing computational time slightly. Therefore, our scheme is more suitable for practical applications.

8. Conclusion

Recently, Xiong et al. proposed an efficient CLAS scheme. They claimed that both of their schemes are provably secure in the random oracle model. In this paper, we propose a new attack against their scheme. To overcome weakness, we also propose an improved CLAS scheme and show our scheme is provable in the random oracle model.

Conflict of Interests

The authors declare that they have no conflict of interests.

References

- [1] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the Eurocrypt '03*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 416–432, 2003.
- [2] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, 2000.
- [3] US Department of Transportation, "National Highway Traffic Safety Administration, vehicle safety communications project," Final Report, US Department of Transportation, Washington DC, USA, 2006.
- [4] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the CRYPTO '84*, vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, Berlin, Germany, 1985.
- [5] S. Al-Riyami and K. Paterson, "Certificateless public key cryptography," in *Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '03)*, vol. 2894 of *Lecture Notes in Computer Science*, pp. 452–473, Springer, Berlin, Germany, 2003.
- [6] J. Baek, R. Safavi-Naini, and W. Susilo, "Certificateless public key encryption without pairing," in *Proceedings of the Industrial Simulation Conference (ISC '05)*, vol. 3650 of *Lecture Notes in Computer Science*, pp. 134–148, Springer, Berlin, Germany, 2005.
- [7] Y. Sun, F. Zhang, and J. Baek, "Strongly secure certificateless public key encryption without pairing," in *Proceedings of the Cryptology and Network Security (CANS '07)*, vol. 4856 of *Lecture Notes in Computer Science*, pp. 194–208, Springer, Berlin, Germany, 2007.
- [8] J. Lai, W. Kou, and K. Chen, "Self-generated-certificate public key encryption without pairing and its application," *Information Sciences*, vol. 181, no. 11, pp. 2422–2435, 2011.
- [9] D. He, Y. Chen, J. Chen, R. Zhang, and W. Han, "A new two-round certificateless authenticated key agreement protocol without bilinear pairings," *Mathematical and Computer Modelling*, vol. 54, no. 11–12, pp. 3143–3152, 2011.
- [10] D. He, J. Chen, and J. Hu, "A pairing-free certificateless authenticated key agreement protocol," *International Journal of Communication Systems*, vol. 25, no. 2, pp. 221–230, 2012.
- [11] D. He, S. Padhye, and J. Chen, "An efficient certificateless two-party authenticated key agreement protocol," *Computers and Mathematics with Applications*, vol. 64, no. 6, pp. 1914–1926, 2012.
- [12] C. Zhou, W. Zhou, and X. Dong, "Provable certificateless generalized signcryption scheme," *Designs, Codes and Cryptography*, 2012.
- [13] W. Liu and C. Xu, "Certificateless signcryption scheme without bilinear pairing," *Ruan Jian Xue Bao/Journal of Software*, vol. 22, no. 8, pp. 1918–1926, 2011.
- [14] D. He, J. Chen, and J. Hu, "A pairing-free certificateless authenticated key agreement protocol," *International Journal of Communication Systems*, vol. 25, no. 2, pp. 221–230, 2012.
- [15] D. He, Y. Chen, and J. Chen, "An efficient secure certificateless proxy signature scheme without pairings," *Mathematical and Computer Modelling*, vol. 57, no. 9–10, pp. 2510–2518, 2013.
- [16] D. He, B. Huang, and J. Chen, "New certificateless short signature scheme," *IET Information Security*, vol. 7, no. 2, pp. 113–117, 2013.
- [17] R. Castro and R. Dahab, "Efficient Certificateless Signatures Suitable for Aggregation," *Cryptology ePrint Archive*, <http://eprint.iacr.org/2007/454>.
- [18] G. Zheng, L. Yu, H. Xuan, and C. Kefei, "Two certificateless aggregate signatures from bilinear maps," in *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD '07)*, vol. 3, pp. 188–193, August 2007.
- [19] Z. Gong, Y. Long, X. Hong, and K. Chen, "Practical certificateless aggregate signatures from bilinear maps," *Journal of Information Science and Engineering*, vol. 26, no. 6, pp. 2093–2106, 2010.
- [20] L. Zhang and F. Zhang, "A new certificateless aggregate signature scheme," *Computer Communications*, vol. 32, no. 6, pp. 1079–1085, 2009.

- [21] X. Hu, Q. Wu, and Z. Chen, "Strong security enabled certificateless aggregate signatures applicable to mobile computation," in *Proceedings of the 3rd IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS '11)*, pp. 92–99, IEEE Computer Society, Washington, DC, USA, December 2011.
- [22] L. Zhang, B. Qin, Q. Wu, and F. Zhang, "Efficient many-to-one authentication with certificateless aggregate signatures," *Computer Networks*, vol. 54, no. 14, pp. 2482–2491, 2010.
- [23] N. Yanai, R. Tso, M. Mambo, and E. Okamoto, "A certificateless ordered sequential aggregate signature scheme secure against super adversaries," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 3, no. 1-2, pp. 30–54, 2012.
- [24] K. Shim, "On the security of a certificateless aggregate signature scheme," *IEEE Communications Letters*, vol. 15, no. 10, pp. 1136–1138, 2011.
- [25] H. Xiong, Z. Guan, Z. Chen, and F. Li, "An Efficient certificateless aggregate signature with constant pairing computations," *Information Science*, vol. 219, no. 10, pp. 225–235, 2013.
- [26] D. He, M. Tian, and J. Chen, "Insecurity of an efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, 2013.
- [27] P. David and S. Jacques, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.