A short certificateless aggregate signature against coalition attacks

Xiaodong Yang, Jinli Wang, Tingchun Ma, Yutong Li, Caifen Wang

College of Computer Science and Engineering, Northwest Normal University, Lanzhou, Gansu, China

Abstract

Certificateless aggregate signature (CLAS) is a crucial cryptosystem. It can not only compress multiple signatures into a short signature, but also ensure the validity of each signature participating in the aggregation by verifying the validity of an resulting aggregate signature. Therefore, a secure and efficient CLAS scheme is very useful for resource-constrained environments because it greatly reduces the overall length of the signature and the verifier’s computational overhead. Cheng et al. presented an efficient CLAS scheme and proved its security in the random oracle model. However, we find that their scheme has security flaws. In this paper, we demonstrate that Cheng et al.’s CLAS scheme is vulnerable to coalition attacks from internal signers. To overcome these attacks, we present an improved CLAS scheme and prove that it is existentially unforgeable under the computational Diffie-Hellman assumption. In addition, our CLAS scheme can not only resist coalition attacks but also generate a very short aggregate signature. The performance analysis results show that our improved CLAS scheme is lower than the related CLAS schemes in terms of communication overhead and computation cost.

1 Introduction

Digital signature is one of the major cryptosystems in modern cryptography, which can authenticate the user’s identity, protect the integrity of the data and achieve the non-repudiation of the user’s behavior. Hence, digital signature is one of the important security technologies in open network environments [1–3], which can ensure data security and identity authentication in new network forms and services [4–6]. In a digital signature scheme, the user uses his/her private key to sign the message, while the corresponding public key is used to verify the legality of the signature generated by the user. However, in a traditional public key cryptosystem, the user’s public key is a random string that cannot be associated with the user’s identity. To authenticate the authenticity of the user’s public key, the user’s identity and public key are usually bound by a public key certificate issued by a certificate authority [7–9]. That is, the validity of the user’s public key is determined by the legitimacy of the corresponding certificate. Unfortunately, the storage and management of numerous public key certificates require
huge computational and storage overheads when deployed in the real world, which greatly
increases the burden of the system.

To eliminate the problem of certificate management in public key cryptosystems, the con-
cept of identity-based cryptography was proposed by Shamir [10]. In an identity-based sig-
nature (IBS) scheme, each user selects a unique identity as his/her public key, such as email
address, telephone number, etc., and the user’s private key is computed by a trusted private
key generator (PKG). Because the user’s public key is a public identity information, it does not
need a certificate to verify the validity of the user’s public key. So, the identity-based signature
scheme implements the binding of the public key and the entity identity in another way, and
greatly simplifies key management. Boneh and Franklin [11] used bilinear pairing to design
the first practical identity-based encryption scheme. Later, many efficient IBS schemes have
also been presented [12–14]. However, the PKG can generate each user’s private key, which
makes the PKG impersonate the user to sign arbitrary message without being discovered.
Hence, key escrow is the main security flaw in identity-based cryptography.

Al-Riyami and Paterson [15] introduced the notion of certificateless signature (CLS) to
solve the inherent key escrow problem in identity-based cryptosystems. In a CLS scheme, the
user’s private key is composed of the secret value and the partial private key. The secret value
is selected by the user himself, and the partial private key is generated by a third party called key
generation center (KGC). Since the KGC does not know the secret value of the user, PKG can-
ot obtain the user’s full private key. In this way, the security threats posed by key escrow are
resolved. In addition, the user’s public key is derived from the secret value but no longer
requires a certificate for authentication. That is, the CLS scheme can basically solve the prob-
lems of both the traditional signature scheme and the IBS scheme. Therefore, CLS has become
one of the most important research hotspots in public key cryptography. The first CLS scheme
was designed by Al-Riyami and Paterson [15], but Huang et al. [16] found that their scheme
[15] was insecure under public key replacement attacks. Subsequently, several efforts have
been made to construct more secure CLS schemes [17–21].

Certificateless aggregate signature (CLAS) has the advantages of CLS and aggregate signa-
ture, which has attracted great attention from researchers. CLAS can compress \( n \) individual
signatures of \( n \) different messages into a short signature. The verifier only needs to verify the
final aggregate signature to determine whether \( n \) individual signatures involved in the aggrega-
tion are valid. As a result, CLAS not only reduces the signature length and the verifier’s com-
putation costs, but also enhances the security of aggregate signature. CLAS is a very useful
technique for reducing communication bandwidth and storage overhead, and it is very suit-
able for environments with limited computing resources and high-efficiency requirements,
such as vehicular ad-hoc networks [22–24], wireless sensor networks [25, 26], the Internet of
things [27], and so on [28, 29].

In recent years, researchers have proposed many CLAS schemes [30–32]. Yum and Lee [33]
presented a CLAS scheme in the random oracle model. However, Hu et al. [34] found that
their CLAS scheme [33] was unable to resist public key replacement attacks. Chen et al. [35]
presented another CLAS scheme, but was vulnerable to public key replacement attacks and
honest-but-curious KGC attacks [36, 37]. Deng et al. [38] constructed an efficient CLAS
scheme and demonstrated its security, but Kumar and Sharma [39] showed that this scheme
[38] could not provide unforgeability. Zhang et al. [40] also proposed an efficient CLAS
scheme and proved its security in the random oracle model. However, Shim [41] pointed out
that their CLAS scheme could not resist collusion attacks from internal users and KGC. Xiong
et al. [42] designed a CLAS scheme with lower signature verification overhead and further con-
structed a CLAS scheme with higher performance. However, Zhang et al. [43] pointed out that
the CLS and CLAS schemes of Xiong et al. [42] could not withstand attacks from a malicious-
but-passive KGC. In addition, Cheng et al. [44] and He et al. [45] noted that the CLS and CLAS schemes of Xiong et al. [42] were also insecure against honest-but-curious KGC attacks, respectively. An improved CLAS scheme was proposed by He et al. [45], but Li et al. [46] demonstrated that this modified scheme was still insecure under malicious-but-passive KGC attacks. Cheng et al. [44] also designed an improved version of Xiong et al.’s CLAS scheme, which was proved to be existentially unforgeable in the random oracle model. Unfortunately, we find that Cheng et al.’s CLAS scheme [44] is also insecure against coalition attacks from the collusion of insider signers. That is, the signers generate multiple invalid individual signatures, but the resulting aggregate signature is valid.

To the best of our knowledge, it is still open and challenging to design a secure and efficient CLAS scheme. In this paper, we focus on this issue, providing a new method for addressing it. We first present a coalition attack of insider signers to show the security flaws of Cheng et al.’s CLAS scheme [44]. Such attacks indicate that the validity of an aggregate signature does not guarantee the legitimacy of each individual signatures participating in the aggregation. However, this result has violated the security objectives of the CLAS scheme. To withstand this type of practical and powerful attack, we further propose an improved CLAS scheme on the basis of Wang et al.’s CLS scheme [47]. In the random oracle model, we prove that the improved CLAS scheme satisfies existential unforgeability under the computational Diffie-Hellman (CDH) assumption. We also demonstrate that our CLAS scheme is secure against coalition attacks. In our CLAS scheme, an aggregate signature is only a group element, so the length of an aggregate signature is independent of the number of signers involved in the aggregation. Additionally, the analysis results indicate that the improved CLAS scheme has lower computational overhead and communication costs.

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries. Section 3 reviews Cheng et al.’s CLAS scheme [44] and gives the corresponding cryptanalysis. Our improved CLAS scheme is presented in Section 4. Section 5 gives the conclusions.

2 Preliminaries

The following are some preliminaries related to this paper, including the security notions of CLAS, bilinear pairing and complexity assumption.

2.1 Formal security model of CLAS

A CLAS scheme mainly includes three entities: KGC, n users and an aggregator, as formally defined in the following seven algorithms [32–34].

- **Setup(λ) → (pp, msk):** Given a security parameter λ, the KGC executes this algorithm to produce the master secret key msk and public parameters pp.
- **PartialKeyGen(pp, msk, IDi) → pski:** Upon input of pp, msk and a user Ui’s identity IDi, the KGC executes this algorithm to produce a partial private key pski for Ui.
- **UserKeyGen(pp, IDi) → (usk, pk):** Upon input of pp and IDi, the user Ui creates its secret value usk and public key pk. Note that the private key of Ui is ski = (usk, pski).
- **Sign(pp, ski, mi) → σi:** Upon input of pp, ski and message mi, the user Ui generates an individual signature σi on mi.
- **Verify(pp, IDi, pk, mi, σi) → {0, 1}:** Upon input of pp, IDi, pk, and a signature σi on a message mi, this algorithm outputs 1 if σi is valid; else, it outputs 0.
• **Aggregate** \( (pp, \{m_i, \sigma_i\}_{i=1}^{n}) \rightarrow \sigma \): Upon input of \( n \) message/signature pairs \( (m_i, \sigma_i) \) from \( n \) users \( U_i \) \((i = 1, \ldots, n)\), the aggregator runs this algorithm to generate an aggregate signature \( \sigma \) on \( \{m_1, \ldots, m_n\} \).

• **AggVerify** \( (pp, \sigma, \{ID_i, pk_i, m_i\}_{i=1}^{n}) \rightarrow \{0, 1\} \): Given \( pp \) and an aggregate signature \( \sigma \) of \( n \) messages \( \{m_1, \ldots, m_n\} \) on \( \{ID_1, \ldots, ID_n\} \) and \( \{pk_1, \ldots, pk_n\} \), this algorithm outputs 1 if \( \sigma \) is valid; else, it outputs 0.

The security model of a CLAS scheme mainly considers two types of attackers [42, 44, 48], named Type I adversary \( A_1 \) and Type II adversary \( A_2 \).

- \( A_1 \) represents a dishonest user (i.e., an external attacker). \( A_1 \) can know the secret value of each user and replace any user’s public key, but does not know the master secret key \( msk \).

- \( A_2 \) represents a malicious KGC. A honest-but-curious KGC knows \( msk \) and generates the user’s private key, but is unable to perform public key replacement. In addition to having the ability of a honest-but-curious KGC, a malicious-but-passive KGC is also allowed to embed trapdoors in \( msk \) and public parameters \( pp \) during the system initialization phase.

The following uses two security games between an attacker \( A \in \{A_1, A_2\} \) and a challenger \( C \) to define the unforgeability of a CLAS scheme.

**Game 1**: The first game is played by \( C \) and \( A_1 \).

- **Initialization**: \( C \) sends public parameters \( pp \) generated by the algorithm \( \text{Setup}(\lambda) \) to \( A_1 \), and then secretly stores the master secret key \( msk \).

- **Partial-private-key-query**: If \( A_1 \) asks for a partial private key of an identity \( ID_n \), then \( C \) executes the algorithm \( \text{PartialKeyGen}(pp, msk, ID_n) \) to produce \( psk \), and returns \( psk \) to \( A_1 \).

- **Public-key-query**: If \( A_1 \) requests a public key of \( ID_n \), then \( C \) performs the algorithm \( \text{UserKeyGen}(pp, ID_n) \) to produce \( pk \), and returns \( pk \) to \( A_1 \).

- **Secret-value-query**: If \( C \) receives an identity \( ID_i \) sent by \( A_1 \), then \( C \) responds to \( A_1 \) with a secret value \( usk_i \) output by the algorithm \( \text{UserKeyGen}(pp, ID_i) \).

- **Public-key-replacement-query**: If \( C \) receives \( ID_i \) and \( pk_i \), then the public key of \( ID_i \) is replaced by \( pk_i \).

- **Sign-query**: When \( A_1 \) asks for a signature of a message \( m_i \) and an identity \( ID_0 \), \( C \) runs the algorithm \( \text{Sign}(pp, sk, m) \) to produce an individual signature \( \sigma_i \) of \( m_i \) and sends it to \( A_1 \).

- **Forgery**: \( A_1 \) finally produces a forgery \( (\{m_i', ID_i', pk_i', \sigma_i'\}_{i=1}^{n}, \sigma') \), where at least one of \( n \) identities \( \{ID_1', \ldots, ID_n'\} \) is the identity \( ID' \) of the target user. Assumed that the \( t \)-th signer’s identity \( ID_t' = ID' \) and that \( \sigma_t' \) is an individual signature of \( (ID_t', pk_t') \) on message \( m_t' \), where \( t \in \{1, \ldots, n\} \). If the following three conditions are satisfied, then \( A_1 \) wins the game.

  1. **Partial-private-key-query** has never received an inquiry about \( ID_t' \).
  2. **Sign-query** has never received an inquiry about \( (ID_t', m_t') \).
  3. \( \sigma' \) is a valid aggregate signature of \( \{ID_t', pk_t'\}_{i=1}^{n} \) on messages \( \{m_i'\}_{i=1}^{n} \).

**Game 2**: The second game is played by \( C \) and \( A_2 \). We consider that the second type of attacker is a malicious-but-passive KGC, which is a strengthened security model.

- **Initialization**: \( A_2 \) sends the master secret key \( msk \) and public parameters \( pp \) generated by the algorithm \( \text{Setup}(\lambda) \) to \( C \).
3.1 Review of Cheng et al.'s CLAS scheme

- Queries: In addition to Partial-private-key-query and Public-key-replacement-query, \( \mathcal{A}_2 \) may initiate other queries defined in Game 1.

- Forgery: \( \mathcal{A}_2 \) finally produces a forgery \( \{(m_i', ID_i', pk_i', \sigma_i')_{i=1}^n, \sigma^*\} \). Assume that the target user’s identity \( ID^* = ID_i' \) and that \( \sigma_i' \) is an individual signature of \( (ID^*, pk^*) \) on message \( m_i' \), where \( t \in \{1, \ldots, n\} \). If the following three conditions are satisfied, then \( \mathcal{A}_2 \) wins the game.

1. Sign-value-query has never received an inquiry about \( ID_i' \).
2. Sign-query has never received an inquiry about \( (ID_i', m_i') \).
3. \( \sigma^* \) is a valid aggregate signature of \( \{(ID_i', pk_i')_{i=1}^n \) on messages \( \{m_i'\}_{i=1}^n \).

**Definition 1.** If there is no polynomial-time attacker \( \mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\} \) who can win in the above two games with a non-negligible probability, then a CLAS scheme is said to be existentially unforgeable under adaptive chosen-message attacks [42, 44, 48].

2.2 Bilinear pairing and complexity assumption

Assume that \( p \) is a prime number, \( G_1 \) and \( G_2 \) are two cyclic groups with order \( p \), and \( g \) is a generator of \( G_1 \). If a map \( e: G_1 \times G_1 \rightarrow G_2 \) satisfies the following conditions, then \( e \) is called a bilinear pairing [43]:

- **Bilinearity:** For all \( \gamma_1, \gamma_2 \in \mathbb{Z}_p \), \( e(g^\gamma_1, g^\gamma_2) = e(g, g)^{\gamma_1\gamma_2} = e(g^{\gamma_1}, g^{\gamma_2}) \).

- **Non-degeneracy:** \( e(g, g) \) is not an identity element \( 1_{G_2} \) in \( G_2 \). That is, \( e(g, g) \neq 1_{G_2} \).

- **Computability:** For any \( \gamma_1, \gamma_2 \in \mathbb{Z}_p \), an algorithm for efficiently calculating \( e(g^\gamma_1, g^{\gamma_2}) \) exists.

Given three elements \( (g, g^a, g^b) \in G_1 \), the CDH problem is to calculate \( g^{ab} \in G_1 \), where the unknown values \( a \) and \( b \) are randomly chosen from \( \mathbb{Z}_p \).

**Definition 2 (CDH assumption).** If the probability of any polynomial-time attacker to solve the CDH problem is negligible, then the CDH problem is called intractable [47, 49].

3 Cryptanalysis of Cheng et al.’s CLAS scheme

3.1 Review of Cheng et al.’s CLAS scheme

To overcome the security flaws in Xiong et al.’s CLAS scheme [42], Cheng et al. [44] presented an improved CLAS scheme that is described as follows.

- **Setup:** Given a security parameter \( \lambda \), the KGC selects two multiplicative cyclic groups \( G_1 \) and \( G_2 \) with prime order \( p \), a bit string \( Q \) of length \( l \), a generator \( g \) in \( G_1 \) and a bilinear pairing \( e: G_1 \times G_1 \rightarrow G_2 \). Then, the KGC randomly selects \( s \in \mathbb{Z}_p^* \) and computes \( \text{Pub} = g^s \). The KGC also picks three hash functions \( H_0: \{0, 1\}^l \rightarrow G_1; H_1: \{0, 1\} \rightarrow G_1 \) and \( H_2: \{0, 1\}^* \rightarrow \mathbb{Z}_p^* \). Finally, the KGC secretly stores the master secret key \( msk = s \) and publishes public parameters \( \text{Pub} = \{ G_1, G_2, p, g, e, Q, \text{Pub}, H_0, H_1, H_2 \} \).

- **PartialKeyGen:** After receiving an identity \( ID \), submitted by a user \( \mathcal{U}_t \), the KGC computes \( \text{psk}_t = H_0(ID)^s \) and sends \( ID_t \)’s partial private key \( \text{psk}_t \) to \( \mathcal{U}_t \) through a secure channel.
In the random oracle model, Cheng et al.’s CLAS scheme [44] was proved to satisfy the security requirements.

- **UserKeyGen**: The user $\mathcal{U}_i$ with identity $ID_i$ randomly selects $x_i \in \mathbb{Z}_p^*$ and computes public key $pk_i = g^{x_i}$. Then, $\mathcal{U}_i$ sets its secret value $usk_i = x_i$ and private key $sk_i = (usk_i, psk_i) = (x_i, H_1(ID_i))$.

- **Sign**: For a message $m_i$, the signer $\mathcal{U}_i$ selects a random value $r_i \in \mathbb{Z}_p^*$ and computes $R_i = g^{r_i}$, $h_i = H_2(m_i, ID_i, pk_i, R_i)$ and $V_i = psk_i \cdot (P_{pub})^{h_i} \cdot H_0(Q)^{h_i+1}$. Then, $\mathcal{U}_i$ outputs $\sigma_i = (R_i, V_i)$ as an individual signature of $m_i$.

- **Verify**: Given $ID_i, pk_i$ and an individual signature $\sigma_i = (R_i, V_i)$ on message $m_i$, a verifier computes $H_0(Q), H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, pk_i, R_i)$ and then verifies the following equation:

$$e(V_i, g) = e(R_i^h, H_1(ID_i), P_{pub}) e(pk_i^h R_i, H_0(Q)).$$

If this equation holds, output 1; otherwise, output 0.

- **Aggregate**: After receiving the message-signature pair $(m_i, \sigma_i = (R_i, V_i))$ from user $\mathcal{U}_i$, for $i = 1, \ldots, n$, the aggregator calculates $V = \prod_{i=1}^n V_i$ and outputs an aggregate signature $\sigma = (R_1, \ldots, R_n, V)$.

- **AggVerify**: Given $n$ identities $\{ID_1, \ldots, ID_n\}$, $n$ public keys $\{pk_1, \ldots, pk_n\}$, $n$ messages $\{m_1, \ldots, m_n\}$ and an aggregate signature $\sigma = (R_1, \ldots, R_n, V)$, a verifier first computes $H_0(Q), H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, pk_i, R_i)$ for $i = 1, \ldots, n$. Then, the verifier verifies the following equation:

$$e(V, g) = e(\prod_{i=1}^n R_i^h, H_1(ID_i), P_{pub}) e(\prod_{i=1}^n pk_i^h R_i, H_0(Q)).$$

If this equation holds, output 1; otherwise, output 0.

### 3.2 Attack on Cheng et al.’s CLAS scheme

In the random oracle model, Cheng et al.’s CLAS scheme [44] was proved to satisfy the security requirement of existential unforgeability. Unfortunately, in this subsection, we demonstrate that Cheng et al.’s CLAS scheme [44] cannot resist coalition attacks from internal signers.

Without loss of generality, suppose that $\{ID_1, pk_1\}$ and $\{ID_2, pk_2\}$ are the identity/public key pairs of the signers $\{\mathcal{U}_i, \mathcal{U}_j\}$, respectively. As a result, we show that $\mathcal{U}_i$ cooperates with $\mathcal{U}_j$ to generate invalid individual signature $\sigma_i$ on message $m_i$ for $i = 1, 2$, whereas the resulting aggregate signature $\sigma$ on $\{m_1, m_2\}$ is valid.

1. $\mathcal{U}_i$ randomly selects $r_i \in \mathbb{Z}_p^*$ and computes $R_i = g^{r_i}$ and $h_i = H_2(m_1, ID_1, pk_1, R_i)$. Then, $\mathcal{U}_i$ sends $(P_{pub})^{h_i} \cdot R_i$ to $\mathcal{U}_j$.

2. $\mathcal{U}_j$ randomly selects $r_j \in \mathbb{Z}_p^*$ and computes $R_j = g^{r_j}$ and $h_j = H_2(m_2, ID_2, pk_2, R_j)$. Then, $\mathcal{U}_j$ sends $(P_{pub})^{h_j} \cdot R_j$ to $\mathcal{U}_i$.

3. $\mathcal{U}_i$ computes $V_i^* = psk_i \cdot (P_{pub})^{h_i} \cdot H_0(Q)^{h_i+1} \cdot h_i$ and sets $\sigma_i^* = (R_i, V_i^*)$ as a signature on $m_1$.

4. $\mathcal{U}_j$ computes $V_j^* = psk_j \cdot (P_{pub})^{h_j} \cdot H_0(Q)^{h_j+1} \cdot h_j$ and sets $\sigma_j^* = (R_j, V_j^*)$ as a signature on $m_2$.

5. $\mathcal{U}_i$ cooperates with $\mathcal{U}_j$ to produce an aggregate signature $\sigma^* = (R_1^*, R_2^*, V^*)$ on messages $\{m_1, m_2\}$, where $V^* = V_i^* \cdot V_j^*$.
Clearly, \( \{ \sigma_1^*, \sigma_2^* \} \) are invalid individual signatures of \( \{ U_1, U_2 \} \) on \( \{ m_1, m_2 \} \), respectively. However, the forged aggregate signature \( \sigma^* = (R_1^*, R_2^*, V^*) \) is valid since

\[
e(V', g) = e(V'_i, V'_2, g)
= e(psk_1(P_{pub})^{h_1} H_6(Q)^{s_1} psk_2 \cdot (P_{pub})^{h_2} H_0(Q)^{s_2} g)
= e(psk_1(P_{pub})^{h_1} H_6(Q)^{s_1} psk_2 \cdot (P_{pub})^{h_2} H_0(Q)^{s_2} g)
= e(\prod_{i=1}^{2} H_1(ID_i), P_{pub}) e(\prod_{i=1}^{2} p_{sk} H_0(Q)).
\]

Therefore, dishonest signers can jointly forge a valid aggregate signature for arbitrary messages. This shows that even if some individual signatures participating in the aggregation are invalid, the resulting aggregate signature is also valid. That is, the validity of every individual signature involved in the aggregation cannot be guaranteed by the legitimacy of the corresponding aggregate signature. So, Cheng et al.’s CLAS scheme [44] is insecure under coalition attacks.

4 Our improved CLAS scheme

In Cheng et al.’s CLAS scheme [44], dishonest signers can use some invalid individual signatures to jointly generate a valid aggregate signature. However, the main reason for the success of such attacks is due to the fact that signers can exchange their individual signature values with each other during the generation of an aggregate signature.

4.1 Construction

In this subsection, we design an enhanced CLAS scheme for preventing the above coalition attack. The following is the description of our CLAS scheme.

- **Setup**: The KGC selects two groups \( G_1 \) and \( G_2 \), which have the same prime order \( p \). The KGC randomly selects a generator \( g \) in \( G_1 \) and a bilinear pairing \( e: G_1 \times G_1 \rightarrow G_2 \). The KGC picks hash functions \( H: \{0, 1\}^* \rightarrow \{0, 1\}^l \) and \( H_1, H_2, H_3: \{0, 1\}^* \rightarrow G_1 \). The KGC also chooses a random value \( s \in Z_p^* \) and calculates \( P_{pub} = g^s \). Finally, the KGC publishes public parameters \( pp = \{G_1, G_2, p, g, e, P_{pub}, H, H_1, H_3\} \) and secretly stores the master secret key \( msk = s \).

- The algorithms **PartialKeyGen** and **UserKeyGen** are the same as those in the CLAS scheme of Cheng et al.

- **Sign**: For a message \( m_i \), the user \( U_i \) uses its secret value \( usk_i = x_i \) and partial private key \( psk_i = H_1(ID_i)^{x_i} \) to calculate \( V_i = H_3(m_i, ID_i, pk_i) \) and

\[
\sigma_i = V_i^{m_i} \cdot psk_i.
\]

Then, \( U_i \) outputs \( \sigma_i \) as an individual signature of \( m_i \).

- **Verify**: Given \( ID_i, pk_i \) and an individual signature \( \sigma_i \) on message \( m_i \), a verifier computes \( H_1(ID_i) \) and \( V_i = H_3(m_i, ID_i, pk_i) \), and then checks the following equation:

\[
e(\sigma_i, g) = e(V'_i, p_{sk}) e(H_i(ID_i), P_{pub}).
\]

If it holds, accept \( \sigma_i \) and output 1; else, reject \( \sigma_i \) and output 0.
4.3 Security analysis

We prove that the improved CLAS scheme is existentially unforgeable under Type I and Type II adversaries. Also, we demonstrate that our CLAS scheme can withstand coalition attacks.

Theorem 1. If the security of our CLAS scheme is broken by a polynomial-time Type I adversary $A_1$, then the CDH problem can be solved.

Proof. Suppose that $A_1$ forges a valid aggregate signature with probability $ε_1$ after making at most $q_i$ random oracle queries to $H_i$ ($i = 1, 3$), $q_{psk}$ partial private key queries, $q_{pk}$ public key queries, $q_{rep}$ public key replacement queries, $q_{usk}$ secret value queries and $q_i$ signing queries, then we can construct an algorithm $C$ to solve the CDH problem. After receiving a random CDH instance $(g, g^a, g^b) ∈ G_1^*$, $C$ will utilize $A_1$’s forgery to calculate the CDH value $g^a b$.

- Initialization: $C$ sets $P_{pub} = g^a$ and runs the algorithm $Setup(λ)$ to produce other parameters. Then, $C$ sends public parameters $pp$ to $A_1$. Note that the master secret key $msk = a$ is unknown to $C$. Moreover, $C$ controls two random oracles to simulate hash functions $H_i$ and $H_3$. To simplify the description, it is assumed that $A_1$ has performed related hash queries before making other inquiries. To avoid collisions and consistently respond to all kinds of queries initiated by $A_1$, $C$ maintains a list $L = (ID_i, psk_i, usk_i, pk_i)$ and two other lists...
\((\mathcal{L}_1, \mathcal{L}_2)\) that are all initially empty. \(C\) picks a random integer \(d \in [1, q_1]\) and responds to all queries initiated by \(A_i\) in the following way.

- **\(H_1\) queries:** Upon receiving the \(i\)-th \(H_1\) inquiry about an identity \(ID_i\), \(C\) first checks whether an entry for \(ID_i\) exists in \(\mathcal{L}_1\). If yes, \(C\) sends the corresponding value \(H_1(ID_i)\) to \(A_i\). Otherwise, \(C\) randomly selects \(y_1 \in \mathbb{Z}_q\) and sets

\[
H_i(ID_i) = \begin{cases} 
g^{y_1} & i \neq d \\
(g^{y_1})^y & i = d 
\end{cases}
\]

If \(i \neq d\), \(C\) adds \((ID_i, y_1, H_1(ID_i))\) to \(\mathcal{L}_1\); otherwise, \(C\) sets the target user’s identity \(ID' = ID_d\) and \(y' = y_d\) and then adds \((ID', y', H_1(ID'))\) to \(\mathcal{L}_1\). Finally, \(C\) sends \(H_1(ID')\) to \(A_i\).

- **\(H_3\) queries:** \(A_i\) issues an \(H_3\) query for \((m_i, ID, pk_i)\). If it is found in \(\mathcal{L}_n\), then \(C\) sends the corresponding value \(V_i\) to \(A_i\). Otherwise, \(C\) randomly selects \(z_i \in \mathbb{Z}_q\) and sets \(V_i = H_3(m_i, ID, pk_i) = g^{z_i}\). Next, \(C\) sends \(V_i\) to \(A_i\) and adds \((m_i, ID, pk_i, z_i, V_i)\) to \(\mathcal{L}_n\).

- **Partial-private-key-query:** After receiving an identity \(ID_i\) submitted by \(A_i\), \(C\) considers the following two cases.
  1. **If \(ID_i = ID'\), \(C\) terminates the game.**
  2. **If \(ID_i \neq ID'\), \(C\) checks whether \(\mathcal{L}\) contains an entry for \(ID_i\).**
     - 1. If it exists, \(C\) sends \(psk_i\) to \(A_i\).
     - 2. Else, \(C\) finds the tuple \((ID_i, y_i, H_1(ID_i))\) in \(\mathcal{L}_1\) and returns a partial private key \(psk_i = (P_{pub})^{y_i}\) to \(A_i\). Then, \(C\) adds \((ID_i, psk_i, \bot, \bot)\) to \(\mathcal{L}\).

- **Public-key-query:** On receiving an identity \(ID_i\), \(C\) checks if an entry for \(ID_i\) exists in \(\mathcal{L}\), where \(pk_i \neq \bot\). If yes, \(C\) sends \(pk_i\) to \(A_i\). Otherwise, \(C\) randomly selects \(x_i \in \mathbb{Z}_p\) and sets \(usk_i = x_i\) and \(pk_i = g^{x_i}\). If \(ID_i = ID'\), \(C\) sets the target user’s public key \(pk' = pk_i\) and secret value \(usk' = usk_i\). Then, \(C\) adds the corresponding tuple to \(\mathcal{L}\) and returns a public key \(pk_i\) to \(A_i\).

- **Secret-value-query:** If \(A_i\) issues a query for an identity \(ID_i\), \(C\) first initiates a public key query about \(ID_i\) and then returns the corresponding secret value \(usk_i\) to \(A_i\).

- **Public-key-replacement-query:** If \(A_i\) submits an identity \(ID_i\) and a public key \(pk_i\), \(C\) first checks whether there is an entry for \(ID_i\) in \(\mathcal{L}\). If yes, \(C\) replaces \(ID_i\)’s public key with \(pk_i\); otherwise, \(C\) sets \(ID_i\)’s public key to \(pk_i\) and then adds a new tuple \((ID_i, \bot, \bot, pk_i)\) in \(\mathcal{L}\).

- **Sign-query:** \(A_i\) asks a signature query on its selected identity \(ID_i\) and message \(m\). If \(ID_i = ID'\), \(C\) terminates the game. Else, \(C\) searches the lists \(\mathcal{L}\) and \(\mathcal{L}_1\) to find the corresponding tuples \((psk_i, usk_i, pk_i)\) and \(V_i = H_3(m_i, ID_i, pk_i)\), respectively. Then, \(C\) returns \(\sigma_i = V_i^{usk_i} \cdot psk_i\) to \(A_i\) as a signature on \(m_i\).

- **Forgery:** \(A_i\) eventually produces a valid forgery \(\{m'_i, ID', pk'_i, \sigma'_i\}_{i=1}^n, \sigma'\). Suppose that the individual signature \(\sigma'_i\) is not returned by **Sign-query** on the target identity \(ID'_i\) and message \(m'_i\), where \(i \in [1, \ldots, n]\). If \(ID'_i \neq ID'_i\) or \(V' = H_3(m'_i, ID', pk')\) is not found in \(\mathcal{L}_n\), \(C\) aborts. Else, \(C\) finds \((usk' = x', pk')\) corresponding to \(ID'_i\) in \(\mathcal{L}\). Then, \(C\) obtains the tuples \((ID', y', H_1(ID'))\) and \((m'_i, ID', pk', z', V')\) from \(\mathcal{L}_1\) and \(\mathcal{L}_n\), respectively. Since \(\sigma'_i\) is valid, the
following equation must hold:
\[
\sigma_i^* = (V^*)^x H_s(ID^s)^a = (V^*)^x (g^{by})^a \\
= (V^*)^x (g^{ab})^y.
\]
From the above equation, C calculates \(g^{ab}\) as
\[
g^{ab} = \left(\frac{\sigma_i^*}{(V^*)^x}\right)^{(y)^{-1}}.
\]
Hence, C solves the CDH problem.

Next, we analyze the successful probability that C can solve the CDH problem instance. If the following events occur, C will not quit during the entire simulation game.

- During the simulation of Partial-private-key-query, each identity \(ID_i\) queried by \(A_1\) is not equal to \(ID^*\). The probability is at least \(\left(1 - \frac{1}{q_1}\right)^{q_{usk}}\).
- During the simulation of Sign-query, each identity \(ID_i\) queried by \(A_1\) is not equal to \(ID^*\). The probability is at least \(\left(1 - \frac{1}{q_1}\right)^{q_1}\).
- In the forgery phase, at least one of \(n\) identities \(\{ID_1, \ldots, ID_n\}\) is the target user’s identity \(ID^*\). The probability of \(ID^* = ID_t\), where \(t \in [1, \ldots, n]\), is at least \(1 - \left(1 - \frac{1}{q_1}\right)^n\).
- \(V^*\) can be found in \(\mathcal{L}_3\). The probability is at least \(\left(1 - \frac{2}{\tilde{t}}\right)\).

Hence, the overall success probability of \(C\) solving the CDH problem is
\[
\epsilon'_1 \geq (1 - \frac{1}{q_1})^{q_{usk}} (1 - \frac{1}{q_1})^{q_1} (1 - (1 - \frac{1}{q_1})^{n}) (1 - \frac{q_1}{P}) \epsilon_1 \\
= (1 - \frac{1}{q_1})^{q_{usk} + q_1} (1 - (1 - \frac{1}{q_1})^{n}) (1 - \frac{q_1}{P}) \epsilon_1.
\]

**Theorem 2.** If the security of our CLAS scheme is broken by a polynomial-time Type II adversary \(A_2\), then the CDH problem can be solved.

**Proof.** Let \(A_2\) be a malicious-but-passive KGC. Suppose that \(A_2\) forges a valid aggregate signature with probability \(\epsilon_2\) after making at most \(q_1\) random oracle queries to \(H_i(i = 1, 3)\), \(q_{pk}\) public key queries, \(q_1\) signing queries and \(q_{msk}\) secret value queries, then there exists a solver \(C\) that invokes \(A_2\) to violate the CDH assumption. Given a CDH instance \((g, g^a, g^b) \in \mathcal{G}_1^3\), the goal of \(C\) is to calculate \(g^{ab}\) by interacting with \(A_2\).

- **Initialization:** \(A_2\) picks a random value \(s \in \mathbb{Z}_p^*\) as the master secret key \(msk\) and calculates \(P_{pub} = g^s\). Then, \(A_2\) runs \(Setup(\lambda)\) to produce other parameters and sends \((pp, msk)\) to \(C\). After that, \(C\) picks a random integer \(d \in [1, q_1]\) and maintains three initially empty lists \(\mathcal{L}, \mathcal{L}_1\) and \(\mathcal{L}_3\).
- **H1 queries:** Upon receiving an \(H_1\) query on an identity \(ID_0\), \(C\) returns the corresponding value \(H_1(ID_0)\) to \(A_2\), if \(\mathcal{L}_1\) contains an entry for \(ID_0\). Otherwise, \(C\) selects \(y_i \in \mathbb{Z}_p^*\) at random and sets \(H_1(ID_0) = g^{y_i}\). Then, \(C\) sends \(g^{y_i}\) to \(A_2\) and adds \((ID_0, y_i, H_1(ID_0))\) to \(\mathcal{L}_1\).
- **H3 queries:** Upon receiving the \(i\)-th \(H_3\) query on \((m_i, ID_n, pk_i)\), \(C\) sends the corresponding \(V_i\) to \(A_2\) if an entry for \((m_i, ID_n, pk_i)\) exists in \(\mathcal{L}_n\). Otherwise, \(C\) selects a random value \(z_i \in \mathbb{Z}_p^*\)
and sets
\[ V_i = H_1(m_i, ID_i, pk_i) = g^{z_i} \begin{cases} i \neq d \\ (g^a)^{z_i} \end{cases} \]

If \( i \neq d \), \( C \) adds \((m_i, ID_i, pk_i, z_i, V_i)\) to \( \mathcal{L}_i \); otherwise, \( C \) sets the target user’s identity \( ID' = ID_i \) and \( pk' = pk_i \) and then adds \((m_i, ID', pk', z_i, V_i)\) to \( \mathcal{L}_i \). Finally, \( C \) sends \( V_i \) to \( A_i \).

- **Public-key-query:** \( A_i \) asks a public key query for \( ID_i \) and if it is found in \( \mathcal{L} \), then \( C \) returns \( pk_i \) to \( A_i \). Otherwise, \( C \) selects a random value \( x_i \in Z_q \) and sets
\[ pk_i = \begin{cases} g^{z_i} & \text{ if } ID_i \neq ID' \\ g^a & \text{ if } ID_i = ID' \end{cases} . \]

If \( ID_i \neq ID' \), \( C \) sets \( usk_i = x_i \) and adds \((m_i, x_i, pk_i)\) to \( \mathcal{L} \). Otherwise, \( C \) sets the public key \( pk' = pk_i \) of the target user, and adds \((ID', b, pk')\) to \( \mathcal{L} \). Finally, \( C \) returns \( pk_i \) to \( A_i \). Note that the target user’s secret value \( usk' = a \), while \( a \) is unknown to \( C \).

- **Secret-value-query:** On receiving the query on \( ID_i \), \( C \) checks if \( ID_i = ID' \); otherwise, \( C \) first initiates a public key query about \( ID_i \) and then returns the corresponding secret value \( usk_i \) to \( A_i \).

- **Sign-query:** Upon receiving an identity \( ID_i \) and a message \( m_i \), \( C \) proceeds as follows:
  - If \( ID_i = ID' \), \( C \) aborts.
  - Else, \( C \) searches the lists \( \mathcal{L} \) and \( \mathcal{L}_i \) to find \((ID_i, x_i, pk_i)\) and \((ID_i, y_i, H_i(ID_i))\), respectively.
    
    Next, \( C \) makes an \( H_i \) query on \((m_i, ID_i, pk_i)\) to obtain \( V_i = H_i(m_i, ID_i, pk_i) \). Finally, \( C \) computes \( \sigma_i = V_i^{s_i} \cdot H_i(ID')^{s_i} \) and returns a signature \( \sigma_i \) on \( m_i \) to \( A_i \).

- **Forgery:** \( A_i \) eventually produces a valid forgery \( \{(m_{i_t}', ID_{i_t}', pk_{i_t}', \sigma_{i_t}')\}_{t=1}^{n} \). Suppose that the individual signature \( \sigma_{i_t}' \) is not returned by **Sign-query** on identity \( ID_{i_t}' \) and message \( m_{i_t}' \), where \( t \in \{1, \ldots, n\} \). If \( ID_{i_t}' \neq ID_i \) or \( V' = H_i(m_{i_t}', ID_{i_t}', pk_{i_t}') \) is not found in \( \mathcal{L}_{i_t} \), \( C \) aborts. Otherwise, \( C \) looks up the lists \( \mathcal{L}_i \) and \( \mathcal{L}_{i_t} \) to obtain the tuples \((ID_{i_t}', y_{i_t}', H_{i_t}(ID_{i_t}'))\) and \((m_{i_t}', ID_{i_t}', pk_{i_t}', z_{i_t}', g^{z_{i_t}'})\), respectively. Since \( \sigma_{i_t}' \) is valid, the following equation must hold:
\[ \sigma_{i_t}' = (V')^s H_{i_t}(ID_{i_t}') = (g^{bc})^{s_i} H_{i_t}(ID_{i_t}') = (g^{ab})^{s_i} H_{i_t}(ID')^{s_i} . \]

Thus, \( C \) can use the master key \( msk = s \) to calculate the CDH value
\[ g^{ab} = (\sigma_{i_t}' / H_{i_t}(ID_{i_t}'))^{(s_i)^{-1}} . \]

Here, we also analyse the probability of \( C \) using \( A_i \)'s forgery to successfully solve the CDH instance. \( C \) completes the entire simulation if the following events occur:

- **ID'** has never appeared during secret value queries. The probability is at least \((1 - \frac{1}{q_i})^{\text{mask}}\).

- **ID'** has never appeared during signing queries. The probability is at least \((1 - \frac{1}{q_i})^{\text{sig}}\).

- In the forgery phase, at least one of \( n \) identities \( \{ID_1', \ldots, ID_n'\} \) is the target user’s identity \( ID' \). The probability of \( ID' = ID_i \), where \( t \in \{1, \ldots, n\} \), is at least \( 1 - (1 - \frac{1}{q_i})^n \).
• $V^*$ can be found in $L_3$. The probability is at least $(1 - \frac{q}{p})$.

Therefore, the overall success probability of $C$ solving the CDH problem is

$$\epsilon' = (1 - \frac{1}{q_3})^{q_3} (1 - \frac{1}{q_3})^{q_3} (1 - (1 - \frac{1}{q_3})^{q_3})(1 - \frac{q}{p})^e_2$$

$$= (1 - \frac{1}{q_3})^{q_3 + q_3} (1 - (1 - \frac{1}{q_3})^{q_3})(1 - \frac{q}{p})^e_2.$$  

Since the CDH problem is intractable, the success probability of $A_1$ or $A_2$ attacking our CLAS scheme is negligible. According to Theorem 1 and Theorem 2, the following theorem is easily obtained.

**Theorem 3.** If the CDH assumption holds, then our improved CLAS scheme satisfies existential unforgeability in the random oracle model.

**Theorem 4.** Our CLAS scheme is secure under coalition attacks.

**Proof.** If an aggregate signature $\sigma$ is valid, then the following equation must hold:

$$\sigma = H(e(V_1, pk_i)e(H_1(ID_i), P_{pub}), \ldots, e(V_n, pk_n)e(H_1(ID_n), P_{pub}))$$

$$= H(e(\sigma_1, g), \ldots, e(\sigma_n, g)).$$

Since the hash function $H$ satisfies collision resistance, we conclude that

$$e(\sigma_i, g) = e(V_i, pk_i)e(H_1(ID_i), P_{pub}), \quad i = 1, \ldots, n.$$  

It shows that each individual signature $\sigma_i$ participating in the aggregation is valid.

Meanwhile, if every individual signature $\sigma_i$ is valid, the following equations hold:

$$e(\sigma_i, g) = e(V_i, pk_i)e(H_1(ID_i), P_{pub}), \quad i = 1, \ldots, n.$$  

Hence, we have

$$\sigma = H(e(\sigma_1, g), \ldots, e(\sigma_n, g))$$

$$= H(e(V_1, pk_i)e(H_1(ID_i), P_{pub}), \ldots, e(V_n, pk_n)e(H_1(ID_n), P_{pub})).$$

This shows that the resulting aggregate signature $\sigma$ generated by $\sigma_i (i = 1, \ldots, n)$ is also valid.

The above analysis indicates that if the internal signers exchange each other's individual signature value during aggregate signature generation, the collision resistance of $H$ ensures that the final aggregate signature could not pass the signature verification equation. In other words, an aggregate signature is valid in our CLAS scheme if and only if each individual signature used to generate the aggregate signature is also valid. Hence, our CLAS scheme can resist coalition attacks.

### 4.4 Performance analysis

We compare our CLAS scheme with other similar schemes [27, 30, 31, 35, 39, 42–46] in terms of performance and security properties. In Table 1, we give the description of some symbols that need to be used in this subsection.

#### 4.4.1 Performance comparison

Because the computational costs of multiplication, hash function and other cryptographic operations are relatively small, we focus on the time-consuming bilinear pairing and exponentiation operations when analysing the performance. Table 2 shows a detailed performance comparison between our CLAS scheme and the related CLAS schemes, where the required symbols are given in Table 1.
From Table 2, for the computational cost of an individual signature generation, our CLAS scheme needs to perform only one exponentiation operation. Zhang et al.’s scheme [43] needs to perform 2 exponentiations. Cheng et al.’s scheme [44] requires 3 exponentiations, as do the schemes of Liu et al. [27], Xiong et al. [42] and He et al. [45]. The other five CLAS schemes [30, 31, 35, 39, 46] need to perform 4 exponentiations.

In our CLAS scheme, the verification equation for an aggregate signature

\[ \sigma = H(e(V_1, pk_1)e(H_1(ID_1), P_{pub}), \ldots , e(V_n, pk_n)e(H_1(ID_n), P_{pub})), \]

where \( e(H_1(ID_i), P_{pub})(i = 1, \ldots , n) \), can be pre-computed. During the aggregate signature verification, our CLAS scheme requires \( n \) bilinear pairings. The scheme of Chen et al. [35] requires 4 bilinear pairings and \( 2n \) exponentiations, as do Du et al.’s scheme [30] and Chen et al.’s scheme [31]. The CLAS scheme in [27, 39, 44–46] requires 3 bilinear pairings and \( 2n \) exponentiations, while Zhang et al.’s scheme [43] requires \( 2n \) bilinear pairings and \( 2n \) exponentiations.

An aggregate signature is only one element of \( G_1 \) in our CLAS scheme and Zhang et al.’s scheme [43]. In Liu et al.’s CLAS scheme [27], an aggregate signature is 2 elements in \( G_1 \), which is the same as in Du et al.’s scheme [30] and Chen et al.’s scheme [35]. In the other six CLAS scenarios [31, 39, 42, 44–46], an aggregate signature consists of \( (n + 1) \) elements in \( G_1 \).

Except for our CLAS scheme and Zhang et al.’s scheme [43], all other CLAS schemes [27, 30, 31, 35, 39, 42, 44–46] are insecure against coalition attacks. However, our CLAS scheme is superior to Zhang et al.’s scheme [43] in terms of computational performance.

### Table 1. Definition of some notations.

| Symbol      | Definition                                      |
|-------------|-------------------------------------------------|
| Size        | the overall length of an aggregate signature     |
| Sign        | the computational cost of the algorithm \( \text{Sign} \) |
| AggVerify   | the computational cost of the algorithm \( \text{AggVerify} \) |
| Coalition attack | Can a scheme resist coalition attacks? |
| \( n \)     | the overall number of signers participating in the aggregation |
| \( |G_1| \) | the length of an element in \( G_1 \) |
| \( P \)     | the computational cost of a bilinear pairing operation |
| \( E \)     | the computational cost of an exponentiation calculation |

https://doi.org/10.1371/journal.pone.0205453.t001

### Table 2. Performance comparison of some CLAS schemes.

| Scheme                     | Sign | AggVerify | Size              | Coalition attack |
|----------------------------|------|-----------|-------------------|------------------|
| Scheme of Cheng et al. [44] | 3E   | 3P+2nE    | \((n + 1)|G_1| \) | No               |
| Scheme of Zhang et al. [43] | 2E   | 2nP+2nE   | \( |G_1| \)          | Yes              |
| Scheme of Liu et al. [27]  | 3E   | 3P+2nE    | 2\(|G_1| \)       | No               |
| Scheme of Xiong et al. [42] | 3E   | 3P+2nE    | \((n + 1)|G_1| \) | No               |
| Scheme of He et al. [45]   | 3E   | 3P+2nE    | \((n + 1)|G_1| \) | No               |
| Scheme of Du et al. [30]   | 4E   | 4P+2nE    | 2\(|G_1| \)       | No               |
| Scheme of Chen et al. [31] | 4E   | 4P+2nE    | \((n + 1)|G_1| \) | No               |
| Scheme of Chen et al. [35] | 4E   | 4P+2nE    | 2\(|G_1| \)       | No               |
| Scheme of Kumar et al. [39] | 4E   | 3P+2nE    | \((n + 1)|G_1| \) | No               |
| Scheme of Li et al. [46]   | 4E   | 3P+2nE    | \((n + 1)|G_1| \) | No               |
| Our scheme                | E    | \( nP \)  | \( |G_1| \)         | Yes              |

https://doi.org/10.1371/journal.pone.0205453.t002
4.4.2 Experimental evaluation. We also conduct performance evaluation experiments on the proposed CLAS scheme and other CLAS schemes [43, 44, 46]. The simulated environment is set up on a laptop with the Windows 10 operating system, and the main configuration is Intel(R) Core(TM) i7-6500 CPU @ 2.59 GHz and 8 GB RAM. To execute the bilinear pairing operation quickly, we choose the curve $a.param$ of type A in the PBC-0.47-VC library. The detailed simulation results are shown in Figs 1, 2 and 3.

Fig 1 shows that each signer needs approximately 26 ms to generate an individual signature in our CLAS scheme. The scheme of Zhang et al. [43] needs approximately 49 ms. Cheng et al.’s scheme [44] requires approximately 52 ms, and Li et al.’s scheme [46] requires approximately 91 ms. Therefore, our CLAS scheme requires the least computational cost when generating an individual signature.

The communication overhead of a CLAS scheme mainly relies on the size of the aggregate signature. Table 2 shows that our CLAS scheme and Zhang et al.’s scheme [43] have the same aggregate signature length, and Cheng et al.’s scheme [44] and Li et al.’s scheme [46] also have the same aggregate signature length. Fig 2 shows that the size of an aggregate signature in our CLAS scheme is fixed at 128 bits, while the length of an aggregate signature in Cheng et al.’s scheme [44] increases linearly with the number of signers. Hence, our CLAS scheme greatly reduces the communication cost.

Fig 3 shows that the computational cost of verifying an aggregated signature is linearly increasing with the total number of signatories. Notably, the verification cost of an aggregate signature is less than the total cost of $n$ individual signature verifications participating in the aggregation. Compared with other CLAS schemes [43, 44, 46], Fig 3 shows that our CLAS scheme has lower computational overhead when verifying aggregate signatures.

In summary, the results of the performance evaluations performed in the above experiments are consistent with the theoretical analysis results in Table 2. From the above analysis,
Fig 2. Comparison of communication cost.

https://doi.org/10.1371/journal.pone.0205453.g002

Fig 3. Time cost of aggregate signature verification.

https://doi.org/10.1371/journal.pone.0205453.g003
we can see that our improved CLAS scheme has lower computational overhead, shorter signature length and higher security.

5 Conclusions

Cheng et al. [44] proposed an efficient CLAS scheme and proved its security. In this paper, we demonstrate that their CLAS scheme [44] cannot resist coalition attacks from internal signers. To prevent these attacks, we design an enhanced CLAS scheme that is secure under coalition attacks. Under the CDH assumption, our improved scheme is proved to satisfy existential unforgeability in the random oracle model. In addition, our CLAS scheme can aggregate \( n \) individual signatures into a short signature which is only an element in \( G_1 \). More importantly, the length of an aggregate signature is fixed and has nothing to do with the total number of signers participating in the aggregation. Therefore, our CLAS scheme is very suitable for network environments with limited resources, such as the Internet of vehicles, and wireless communication networks [50–52].

Supporting information

S1 Data. Time cost of individual signature generation. (XLSX)
S2 Data. Comparison of communication cost. (XLSX)
S3 Data. Time cost of aggregate signature verification. (XLSX)

Author Contributions

Data curation: Tingchun Ma.
Formal analysis: Yutong Li.
Validation: Jinli Wang.
Writing – original draft: Xiaodong Yang.
Writing – review & editing: Caifen Wang.

References

1. Sun X, Wang H, Li J, Zhang Y (2011) Injecting purpose and trust into data anonymisation. Computers & Security, 30(5): 332–345. https://doi.org/10.1016/j.cose.2011.05.005
2. Wang H, Cao J, Zhang Y (2002) Ticket-based service access scheme for mobile users. In: Proc. Australian Computer Science Communications. pp. 285-292.
3. Sun X, Wang H, Li J, Zhang Y (2012) Satisfying privacy requirements before data anonymization. The Computer Journal, 55(4): 422–437. https://doi.org/10.1093/comjnl/bxr028
4. Shen Y, Zhang T, Wang Y, Wang H, Jiang X (2017) MicroThings: A generic IoT Architecture for flexible data aggregation and scalable service cooperation. IEEE Communications Magazine 55(9): 86–93. https://doi.org/10.1109/MCOM.2017.1700104
5. Sun X, Li M, Wang H, Plank A (2008) An efficient hash-based algorithm for minimal k-anonymity. In: Proc. the Thirty-First Australasian Conference on Computer Science. pp. 101-107.
6. Zhang Y, Shen Y, Wang H, Zhang Y, Jiang X (2018). On secure wireless communications for service oriented computing. IEEE Transactions on Services Computing 11(2): 318–326. https://doi.org/10.1109/TSC.2015.2478453
7. Szepieniec A, Beullens W, Preneel B (2017) MQ signatures for PKI. In: Proc. International Workshop on Post-Quantum Cryptography. pp. 224-240.
8. Al-Bassam M (2017) SCPKI: a smart contract-based PKI and identity system. In: Proc. the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. pp. 35-40.
9. Li X, Niu J, Kumari S, Wu F, Sangaiah AK, Choo KKR (2018) A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments. Journal of Network and Computer Applications 103: 194–204. https://doi.org/10.1016/j.jnca.2017.07.001
10. Shamir A (1984) Identity-based cryptosystems and signature schemes. In: Proc. Workshop on the Theory and Application of Cryptographic Techniques. pp. 47-53.
11. Boneh D and Franklin M (2003) Identity-based encryption from the Weil pairing. SIAM Journal on Computing, 32(3): 586–615. https://doi.org/10.1137/S0097539701398521
12. Wei J, Liu W, Hu X (2017) Forward-secure identity-based signature with efficient revocation. International Journal of Computer Mathematics, 94(7): 1390–1411. https://doi.org/10.1080/00207160.2016.1190835
13. Shen L, Ma J, Liu X, Wei F, Miao M (2017) A secure and efficient id-based aggregate signature scheme for wireless sensor networks. IEEE Internet of Things Journal 4(2): 546–554. https://doi.org/10.1109/JIOT.2016.2557487
14. Jia X, He D, Zeadally S, Li L (2017) Efficient revocable id-based signature with cloud revocation server. IEEE Access, 5: 2945–2954. https://doi.org/10.1109/ACCESS.2017.2676021
15. Al-Riyami SS, Paterson KG (2003) Certificateless public key cryptography. In: Proc. International Conference on the Theory and Application of Cryptography and Information Security. pp. 452-473.
16. Huang X, Susilo W, Mu Y, Zhang F (2005) On the security of certificateless signature schemes from Asiacrypt 2003. In: Proc. International Conference on Cryptology and Network Security. pp. 13-25.
17. Wang L, Chen K, Long Y, Wang H (2017) An efficient pairing-free certificateless signature scheme for resource-limited systems. Science China Information Sciences 60(11): 1–3.
18. Yuan Y, Wang C (2014) Certificateless signature scheme with security enhanced in the standard model. Information Processing Letters 114(9): 492–499. https://doi.org/10.1016/j.ipl.2014.04.004
19. Hung YH, Huang SS, Tseng YM, Tsai TT (2015) Certificateless signature with strong unforgeability in the standard model. Informatica 26(4): 663–684. https://doi.org/10.15388/Informatica.2015.70
20. Canard S, Trinh VC (2016) An Efficient certificateless signature scheme in the standard model. In: Proc. International Conference on Information Systems Security. pp. 175-192.
21. Hung YH, Tseng YM, Huang SS (2017) Lattice-based revocable certificateless signature. Symmetry 9(10): 1–17. https://doi.org/10.3390/sym9100242
22. Cui J, Zhang J, Zhong H, Shi R, Xu Y (2018) An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. Information Sciences 451: 1–15.
23. Malhi AK, Batra S (2015) An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks. Discrete Mathematics and Theoretical Computer Science 17(1): 317–338.
24. Jin C, Zhao J (2018) Certificateless aggregate deniable authentication protocol for ad hoc networks. International Journal of Electronic Security and Digital Forensics 10(2): 168–187. https://doi.org/10.1504/IJESDF.2018.10010114
25. Kumar P, Kumari S, Sharma V, Sangaiah AK, Wei J, Li X (2017) A certificateless aggregate signature scheme for healthcare wireless sensor network. Sustainable Computing: Informatics and Systems.
26. Shen L, Ma J, Liu X, Miao M (2016) A provably secure aggregate signature scheme for healthcare wireless sensor networks. Journal of medical systems 40(11): 1–10. https://doi.org/10.1007/s10916-016-0613-3
27. Liu J, Han J, Wu L, Sun R, Du X (2017) VDAS: verifiable data aggregation scheme for Internet of Things. In: Proc. 2017 IEEE International Conference on Communications. pp. 1-6.
28. Deng J, Xu C, Wu H, Dong L (2016) A new certificateless signature with enhanced security and aggregation version. Concurrency and Computation: Practice and Experience 28(4): 1124–1133. https://doi.org/10.1002/cpe.3551
29. Zhang L, Qin B, Wu Q, Zhang F (2010) Efficient many-to-one authentication with certificateless aggregate signatures. Computer Networks 54(14): 2482–2491. https://doi.org/10.1016/j.comnet.2010.04.008
30. Du HZ, Huang MJ, Wen QY (2013) Efficient and provably-secure certificateless aggregate signature scheme. Dianzi Xuebao(Acta Electronica Sinica) 41(1): 72–76.
31. Chen H, Wei S, Zhu C, Yang Y (2015) Secure certificateless aggregate signature scheme. Journal of Software 26(5): 1173–1180.
32. Liu H, Liang M, Sun H (2014) A secure and efficient certificateless aggregate signature scheme. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 97(4): 991–995. https://doi.org/10.1587/transfun.E97.A.991
33. Yum DH, Lee PJ (2004) Generic construction of certificateless signature. In: Proc. Australasian Conference on Information Security and Privacy. pp. 200-211.
34. Hu BC, Wong DS, Zhang Z, Deng X (2006) Key replacement attack against a generic construction of certificateless signature. In: Proc. In Australasian Conference on Information Security and Privacy. pp. 235-246.
35. Chen YC, Tso R, Mambo M, Huang K, Horng G (2015) Certificateless aggregate signature with efficient verification. Security and Communication Networks 8(13): 2232–2243. https://doi.org/10.1002/sec.1166
36. Zhang H (2016) Insecurity of a certificateless aggregate signature scheme. Security and Communication Networks 9(11): 1547–1552. https://doi.org/10.1002/sec.1447
37. Shen H, Chen J, Shen J, He D (2016) Cryptanalysis of a certificateless aggregate signature scheme with efficient verification. Security and Communication Networks 9(13): 2217–2221.
38. Deng J, Xu C, Wu H, Dong L (2016) A new certificateless signature with enhanced security and aggregation version. Concurrency and Computation: Practice and Experience 28(4): 1124–1133. https://doi.org/10.1002/cpe.3551
39. Kumar P, Sharma V (2017) A comment on efficient certificateless aggregate signature scheme. In: Proc. 2017 IEEE Computing, Communication and Automation. pp. 515-519.
40. Zhang L, Zhang F (2009) A new certificateless aggregate signature scheme. Computer Communications 32(6): 1079–1085. https://doi.org/10.1016/j.comcom.2008.12.042
41. Shim KA (2011) On the security of a certificateless aggregate signature scheme. IEEE Communications Letters 15(10): 1136–1138. https://doi.org/10.1109/LCOMM.2011.081011.111214
42. Xiong H, Guan Z, Chen Z, Li F (2013) An efficient certificateless aggregate signature with constant pairing computations. Information Sciences 219: 225–235. https://doi.org/10.1016/j.ins.2012.07.004
43. Zhang F, Shen L, Wu G (2014) Notes on the security of certificateless aggregate signature schemes. Information Sciences 287: 32–37. https://doi.org/10.1016/j.ins.2014.07.019
44. Cheng L, Wen Q, Jin Z, Zhang H, Zhou L (2015) Cryptanalysis and improvement of a certificateless aggregate signature scheme. Information Sciences 295: 337–346. https://doi.org/10.1016/j.ins.2014.09.065
45. He D, Tian M, Chen J (2014) Insecurity of an efficient certificateless aggregate signature with constant pairing computations. Information Sciences 268: 458–462. https://doi.org/10.1016/j.ins.2013.09.032
46. Li J, Yuan H, Zhang Y (2018) Cryptanalysis and improvement for certificateless aggregate signature. Fundamenta Informaticae 157(2): 111–123. https://doi.org/10.3233/FI-2018-1620
47. Wang ZHH, Han ZH, Liu J, Zhang DW, Chang L (2016) ID authentication scheme based on PTPM and certificateless public key cryptography in cloud environment. Journal of Software 27(6): 1523–1537.
48. Au MH, Mu Y, Chen J, Wong DS, Liu JK, Yang G (2007) Malicious KGC attacks in certificateless cryptography. In: Proc. the 2nd ACM Symposium on Information, Computer and Communications Security. pp. 302-311.
49. Yang X, Chen C, Ma T, Wang J, Wang C (2018) Revocable identity-based proxy re-signature against signing key exposure. PLoS ONE 13(3): e0194783. https://doi.org/10.1371/journal.pone.0194783. PMID: 29579125
50. Ogundoyin SO, Awoyemi SO (2018) EDAS: efficient data aggregation scheme for Internet of things. Journal of Applied Security Research 13(3): 347–375. https://doi.org/10.1080/19361610.2018.1463139
51. Kumar P, Kumari S, Sharma V, Li X, Sangaiah AK, Islam SKH (2018) SecureCLS and CL-AS schemes designed for VANETs. The Journal of Supercomputing, 2018: 1–23. https://doi.org/10.1007/s11227-018-2312-y.
52. Tian M, Huang L (2013) Cryptanalysis of a certificateless signature scheme without pairings. International Journal of Communication Systems, 26(11): 1375–1381. https://doi.org/10.1002/dac.2310