Hindawi
Security and Communication Networks
Volume 2020, Article ID 5276813, 12 pages
https://doi.org/10.1155/2020/5276813

Research Article

Efficient Certificateless Aggregate Signature Scheme for Performing Secure Routing in VANETs

Zhiyan Xu,[1,2] **Debiao He**,[2,3] **Neeraj Kumar**,[4,5,6] **and Kim-Kwang Raymond Choo**[7]

[1]*Hubei Co-Innovation Center of Basic Education Information Technology Services, College of Computer, Hubei University of Education, Wuhan, China*
[2]*Guangdong Provincial Key Laboratory of Data Security and Privacy Protection, Guangzhou, China*
[3]*Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China*
[4]*Department of Computer Science and Engineering, Thapar University, Patiala, India*
[5]*School of Communication and Information Engineering, Asia University, Taichung City, Taiwan*
[6]*King Abdul Aziz University, Jeddah, Saudi Arabia*
[7]*Department of Information Systems and Cyber Security, Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX, USA*

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

Received 20 September 2019; Accepted 6 January 2020; Published 12 February 2020

Academic Editor: A. Peinado

Certificateless public key cryptosystem solves both the complex certificate management problem in the public key cryptosystem based on the PKI and the key escrow issue in the public key cryptosystem based on identity. The aggregator can compress n different signatures with respect to n messages from n signers into an aggregate signature, which can help communication equipments to save a lot of bandwidth and computing resources. Therefore, the certificateless aggregate signature (CLAS) scheme is particularly well suited to address secure routing authentication issues in resource-constrained vehicular ad hoc networks. Unfortunately, most of the existing CLAS schemes have problems with security vulnerabilities or high computation and communication overheads. To avoid the above issues and better solve the secure routing authentication problem in vehicular ad hoc networks, we present a new CLAS scheme and give the formal security proof of our scheme under the CDH assumption in the random oracle model. We then evaluate the performance of our proposed CLAS scheme, and the results demonstrate that our proposal is more practical in resource-constrained vehicular ad hoc networks.

1. Introduction

Vehicular ad hoc networks (VANETs) have drawn comprehensive attention in recent years as they help enhance driving safety and optimize transportation systems [1, 2]. Figure 1 shows a typical VANET architecture in a vehicle-road cooperative system. The VANET is based on sensor detection and wireless communication technology to obtain vehicle road information, which is usually composed of road trusted authorities (TAs), road side units (RSUs) along the roads, and on-board units (OBUs) installed in the vehicles. Through vehicle-vehicle and vehicle-road information exchange and sharing, the traffic control center can effectively understand the traffic environment, further realize the intelligent cooperation between vehicles and the infrastructure, and finally achieve the goal of optimizing system resources and improving road traffic.

Just as everything has two sides, VANETs are bringing convenience to people's lives while also facing great security challenges. On the one hand, mobile VANETs, by exchanging information between vehicles, can enhance the safety of the vehicle and thus ensure passenger safety. On the other hand, dynamic topology and lack of centralized management features make it difficult for users to identify nodes that have malicious behavior in VANETs. VANETs may suffer from malicious attacks such as message tampering, false message sending, and denial of service (DOS) [3]. This means that, in a VANET, there may be malicious

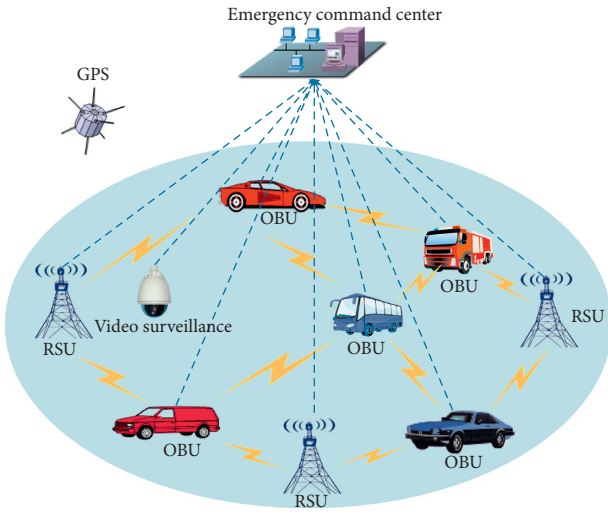

Figure 1: A typical VANET architecture.

nodes broadcasting false information to other nodes and attempting to disrupt route discovery or data transmission, and because the VANET is a network where vehicles are constantly changing positions, secure routing is essential.

To defend against the intruder's attack, VANETs' security design should meet the security attributes such as authenticity, privacy, and integrity. Among them, authenticity can ensure the reliability of a message by correctly identifying the identity of the sender, which can be used to solve the problem of secure routing authentication. Privacy refers to the private communication by using pseudonyms between communication entities. Integrity is the mechanism by which information cannot be tampered with or discarded as it is transmitted from the sender to the receiver. The above attributes are important factors that enable the public to accept and successfully deploy VANETs.

Digital signature [4, 5] can provide routing authentication, integrity detection, and nonrepudiation. Anyone should be able to verify the validity of the signature through the signer's public key, which helps to achieve efficient and secure communication between nodes in the VANET. However, RSUs and traffic control centers in the VANET (generally verified by TAs) all need to verify a large number of route-related signatures in high-density communication scenarios [2], which will result in higher computational burden for nodes, especially for the node in resource-constrained networks. In these situations, it is best to limit the digital signature's communication requirements (i.e., size). One accepted solution is the aggregate signature technology which is the best choice to solve the above problems.

Because the aggregate signature can reduce the node authentication overhead and the certificateless cryptosystem can solve certificate management and key escrow problems existing in the traditional cryptosystem, many researchers combine certificateless cryptography and aggregate signature to further propose various CLAS schemes. The CLAS can not only prevent the routing information from being forged, tampered, and impersonated but also ensure the integrity of the routing information and provide authentication and nonrepudiation for the routing information sender. In this paper, we put forward a CLAS scheme for the practical application environment of the VANETs.

1.1. Our Research Contributions. In this paper, we put forward a novel CLAS scheme which could better support the reliable routing information delivery in the highly dynamic VANETs. The main contributions of this paper are summarized as follows:

(i) Firstly, we define a typical VANET architecture for an emergency linkage scheduling environment, which is more close to the actual application scenario.

(ii) Secondly, we present a CLAS scheme for VANETs, and our new scheme can provide secure routing for VANETs while meeting the security requirements.

(iii) Finally, we prove the security and evaluate the performance of the newly proposed CLAS scheme.

1.2. Organization of the Paper. The remainder of this paper is organized as follows: Section 2 describes the related work. Section 3 gives the problem statement related to our paper, and then we present details of the proposed CLAS scheme in Section 4. Furthermore, in Sections 5 and 6, the security proof and the performance analysis are presented. Finally, we give the conclusion of our work in Section 7.

2. Related Works

To achieve the identity authentication of the message sender and then establish a trust relationship between the nodes, many digital signature schemes have been put forward successively. In a traditional PKI-based public key cryptosystem [6–8], each user has a key pair, public key, and private key, where the former remains public and the latter remains secret. To ensure the correspondence between the user's public key and his/her identity, the certificate authority (CA) needs to issue and maintain a certificate for the user, which involves various certificate management issues such as certificate distribution, storage, and revocation.

In the identity-based public key cryptosystem (ID-PKC) [9–12], the public key is selected by the user himself/herself, and the user's private key is produced by the private key generator (PKG) based on his/her identity information. Because no certificate is required, the ID-PKC can eliminate the problem of certificate management in the PKI. However, since the PKG can obtain any user's private key, the ID-PKC suffers from a key escrow issue which means it must be fully trusted by all users, and this assumption is too strong in some applications.

To solve the problems existing in the above two cryptosystems, researchers in [13] first put forward a certificateless public key cryptosystem (CL-PKC). In the CL-PKC, the user's public key is produced by the user himself/herself, and the user's full private key is generated by the cooperation between the KGC and the user. The former is responsible for

generating the partial private key based on the user's identity, and the latter is responsible for generating the secret value. Therefore, CL-PKC can not only solve the complex certificate management problem in the PKI-based cryptography but also solve the inherent key escrow issue in the identity-based cryptography [14].

The advantages of the CL-PKC have aroused the enthusiasm of researchers, and many certificateless signature (CLS) schemes have been proposed [2, 15, 16]. Huang et al. [15] demonstrated that the CLS scheme proposed in [13] could not resist the public key replacement attack and further proposed an improved CLS scheme. Yum and Lee [2] introduced a generic CLS construction. However, Hu et al. [16] indicated that their scheme is insecure and further proposed an improved CLS scheme. Au et al. [17] proposed an enhanced security model that allows the malicious KGC to produce key pairs in any way. Nevertheless, the certificateless signature schemes proposed in [18, 19] have been found to be insecure against malicious KGC attacks.

Boneh et al. [20] proposed the concept of aggregate signature in Eurocrypt 2003. The aggregator can compress n different signatures with respect to n messages from n different signers into an aggregate signature. The verifier can authenticate the multiple senders simply by verifying the short aggregate signature, which can save the bandwidth and computational cost of mobile devices in VANETs. Because aggregate signatures greatly shorten the length of the signature, they are especially suitable for applications in resource-constrained VANETs.

Gong et al. [21] combined the certificateless public-key cryptosystem with the aggregate signature and then proposed the first CLAS scheme, but they did not present the formal security proof of the scheme. After the groundbreaking work [21], many CLAS schemes [22–27] were proposed for various practical application scenarios. Zhang and Zhang [22] redefined the concept and security model for the CLAS scheme and proposed a new CLAS scheme, but their scheme has been proven to not resist malicious KGC attacks.

Xiong et al. [23] proposed a CLAS scheme, but He et al. [24] found that their scheme was falsifiable and further put forward a new CLAS scheme. The researchers [26, 27] have found that the CLAS scheme proposed in [25] is insecure for malicious KGC attacks. Horng et al. [28] proposed a CLAS scheme, but we found that the scheme cannot resist any type of adversary in the certificateless security model and the signature is falsifiable. More recently, Li et al. [29] demonstrated that there is a security defect in the CLAS scheme proposed in [24] and further put forward an improved CLAS scheme.

3. Problem Statement

In this section, we first describe the bilinear map and relational difficult problems and then introduce the system model of our proposed CLAS scheme. Finally, the system components of the CLAS scheme are given.

3.1. Bilinear Map. Suppose that G_1 and G_2 are two cyclic groups, where prime number q is the order of G_1 and G_2 and P is the generator of G_1. e: $G_1 \times G_1 \longrightarrow G_2$ is a bilinear map. For all $P, Q, S \in G_1$, $a, b \in Z_q^*$, and e should satisfy the following properties:

(1) Bilinearity: $e(P, Q + S) = e(P, Q)e(P, S)$ and $e(aP, bQ) = e(abP, Q) = e(P, abQ)$.

(2) Nondegeneracy: there exists $P \in G_1$ such that $e(P, P) \neq 1$.

(3) Computability: there exists an efficient algorithm to calculate $e(P, Q)$.

3.2. Complexity Assumption

3.2.1. Computational Diffie–Hellman (CDH) Problem. Given a generator P of an additive cyclic group G_1 with the order q and a random instance (aP, bP), it is difficult to calculate abP, where a and b remain unknown.

3.2.2. Computational Diffie–Hellman (CDH) Assumption. There does not exist adversary A, and the CDH problem can be decided in probabilistic polynomial time with a non-negligible probability ε, where $\varepsilon > 0$ is a very small number.

3.3. System Model. In this paper, we take the application of VANETs in the emergency linkage scheduling (ELS) environment as an example and give the corresponding VANET architecture that is shown in Figure 2. There are six types of entities in the VANET architecture: on-board unit (OBU), road side unit (RSU), key generation center (KGC), emergency command center (ECC), signature aggregator (SA), and trusted authority (TA). The entities are specifically defined as follows.

3.3.1. On-Board Unit. On-board unit is a device installed in the vehicle. Let ID_i denote the identity and (SK_i, PK_i) denote the key pair of an OBU. Each OBU can use its private key to generate a signature for the relevant routing information and then send the signature to the signature aggregator.

3.3.2. Road Side Unit. Road side unit is a device installed on the side of the road, which can generate signatures for related messages, realize the exchange and sharing of the vehicle-road information, and further provide local real-time traffic information to the emergency command center.

3.3.3. Key Generator Center. Key generator center is a device that is responsible for generating system parameters params and the partial private key D_i for each OBU or RSU corresponding to his/her identity and then secretly sends D_i to the OBU or RSU.

3.3.4. Emergency Command Center. Emergency command center is a device with strong computing power and plenty of storage space, which can obtain information on the accident scene and surrounding road conditions from OBUs or RSUs

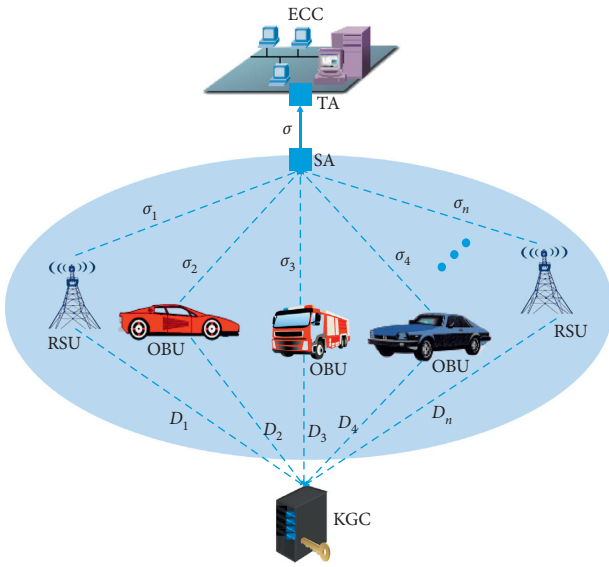

Figure 2: Architecture of a VANET in the ELS environment.

through the vehicle network emergency linkage system and further give corresponding emergency measures to improve rescue efficiency.

3.3.5. Signature Aggregator. Signature aggregator refers to a certain computing power of a device. It is responsible for collecting a single route-related signature from OBUs or RSUs and then generating an aggregate signature and sending it to the corresponding TA.

3.3.6. Trusted Authority. Trusted authority is a device with a certain computing power. It is responsible for verifying the route-related aggregate signature and then outputting a verification result.

3.4. System Components. Our CLAS scheme for performing secure routing in VANETs is a collection of the following seven polynomial time algorithms:

(i) Setup $(1^k) \longrightarrow (\text{params}, s, P_{\text{pub}})$ is a probabilistic algorithm executed by the KGC, where k is the security parameter, params is the system parameter list, s is the system master key, and P_{pub} is the system master public key.

(ii) Partial-Key-Gen $(\text{params}, ID_i) \longrightarrow D_i$ is a probabilistic algorithm executed by the KGC, where params is the system parameter list, $ID_i \in \{0,1\}^*$ is a user's identity, and D_i is the partial private key corresponding to the user's identity ID_i.

(iii) User-Key-Gen $(\text{params}, D_i) \longrightarrow (SK_i, PK_i)$ is a randomized algorithm executed by the user with identity ID_i, where params is the system parameter list, D_i is the partial private key corresponding to the identity ID_i, and (SK_i, PK_i) is the key pair of the user with the identity ID_i.

(iv) Sign $(\text{params}, (SK_i, PK_i), ID_i, m_i) \longrightarrow \sigma_i$ is a randomized algorithm executed by the signer, where params is the system parameter list, (SK_i, PK_i) is the key pair of the signer, ID_i is the signer's identity, m_i is the message, and σ_i is the signature on the message m_i.

(v) Verify $(\text{params}, ID_i, m_i, PK_i, \sigma_i) \longrightarrow \{\text{“0”}, \text{“1”}\}$ is a probabilistic algorithm executed by the verifier, where params is the system parameter list, ID_i is the signer's identity, PK_i is the public key of the signer, m_i is the message, and σ_i is the signature on the message m_i; 1 or 0 is the output to indicate whether the signature σ_i is validated.

(vi) Aggregate $(\text{params}, ID_i, m_i, PK_i, \sigma_i)_{1 \le i \le n} \longrightarrow \sigma$ is a deterministic algorithm executed by the signature aggregator, where params is the system parameter list, ID_i is the signer's identity, PK_i is the public key of the signer, m_i is the message, and σ_i is the signature on the message m_i.

(vii) Aggregate-Verify $(\text{params}, ID_i, m_i, PK_i, \sigma)_{1 \le i \le n} \longrightarrow \{\text{“0”}, \text{“1”}\}$ is a deterministic algorithm executed by the aggregate verifier, where params is the system parameter list and σ is the aggregate signature on the message m_i with the identity ID_i and the public key PK_i. 1 or 0 is the output to indicate whether the aggregate signature σ is validated.

4. Our Proposed CLAS Scheme

To improve the security of routing information in VANETs, we propose a new CLAS scheme. Compared to previous works, our new scheme strives to achieve the following two goals: (1) to ensure the unforgeability of the signature scheme and (2) to improve the performance of the scheme. Our CLAS scheme includes seven phases: Setup, Partial-Key-Gen, User-Key-Gen, Sign, Verify, Aggregate, and Aggregate-Verify. The scheme details are described below.

4.1. Setup. The KGC generates system parameters after obtaining the security parameter k by executing the following operations:

(1) The KGC generates two cyclic groups G_1 and G_2 with the order q, where q is a prime number. P is a generator of G_1. $e: G_1 \times G_1 \longrightarrow G_2$ is a bilinear pairing.

(2) The KGC randomly selects $s \in Z_q^*$ as the master key and calculates $P_{\text{pub}} = sP$ as the public key.

(3) The KGC selects four hash functions: $H_1: \{0,1\}^* \longrightarrow G_1$, $H_2: \{0,1\}^* \longrightarrow G_1$, $h_1: \{0,1\}^* \longrightarrow Z_q^*$, and $h_2: \{0,1\}^* \longrightarrow Z_q^*$.

(4) The KGC maintains s secret and params $= (G_1, G_2, P, e, q, P_{\text{pub}}, H_1, H_2, h_1, h_2)$ public.

4.2. Partial-Key-Gen. The KGC produces the user's partial private key by executing the following operations:

(1) Given ID_i as a user's identity, the KGC first calculates $Q_i = H_1(ID_i)$ and then computes the user's partial private key $D_i = s \cdot Q_i$.

(2) The KGC secretly sends D_i to the corresponding user.

4.3. User-Key-Gen. A user with the identity ID_i generates his/her full private key and public key by executing the following operations:

(1) The user randomly selects $x_i \in Z_q^*$ as the secret value.

(2) The user sets $SK_i = \{D_i, x_i\}$ as a user's full private key.

(3) The user computes $PK_i = x_i P$ as a user's public key.

4.4. Sign. A signer with the identity ID_i produces a signature σ_i on the message m_i by executing the following operations:

(1) The signer inputs system parameters params, signature key pairs (SK_i, PK_i), and the message m_i.

(2) The signer selects $w_i \in Z_q^*$ randomly and then computes $W_i = w_i P$.

(3) The signer computes $\alpha_i = h_1(ID_i, PK_i, W_i)$, $\beta_i = h_2(m_i, ID_i, PK_i, W_i)$, and $Z = H_2(P_{\text{pub}})$.

(4) The signer computes $V_i = \alpha_i D_i + (w_i + \beta_i x_i)Z$.

(5) The signer outputs $\sigma_i = (W_i, V_i)$ as the signature on the message m_i.

4.5. Verify. The verifier verifies the signature $\sigma_i = (W_i, V_i)$ on the message m_i with identity ID_i by executing the following operations:

(1) The verifier computes $\alpha_i = h_1(ID_i, PK_i, W_i)$, $\beta_i = h_2(m_i, ID_i, PK_i, W_i)$, $Q_i = H_1(ID_i)$, and $Z = H_2(P_{\text{pub}})$.

(2) The verifier verifies the following:

$$e(V_i, P) = e\left(\alpha_i Q_i, P_{\text{pub}}\right) e(W_i + \beta_i PK_i, Z). \tag{1}$$

(3) If equation (1) holds, it emits 1 and the verifier accepts σ_i; otherwise, it emits 0 and the verifier rejects σ_i.

4.6. Aggregate. The aggregator generates the aggregate signature σ from user-message-public key-signature pairs $(ID_i, m_i, PK_i, \sigma_i)_{1 \le i \le n}$ by executing the following operations:

(1) The aggregator inputs n tuples $(ID_i, m_i, PK_i, \sigma_i)$, where $1 \le i \le n$.

(2) The aggregator computes $V = \sum_{i=1}^{n} V_i$.

(3) The aggregator outputs $\sigma = (W, V)$ as the aggregate signature, where $W = \{W_1, W_1, \ldots, W_n\}$.

4.7. Aggregate-Verify. The aggregate verifier verifies the validity of the aggregate signature $\sigma = (W, V)$ by executing the following operations:

(1) The aggregate verifier inputs the tuples $(ID_i, m_i, PK_i, \sigma_i)_{1 \le i \le n}$ and the aggregate signature $\sigma = (W, V)$.

(2) The aggregate verifier computes $Z = H_2(P_{\text{pub}})$; furthermore, for $1 \le i \le n$, the aggregate verifier computes $Q_i = H_1(ID_i)$, $\alpha_i = h_1(ID_i, PK_i, W_i)$, and $\beta_i = h_2(m_i, ID_i, PK_i, W_i)$.

(3) The aggregate verifier verifies the following:

$$e(V, P) = e\left(\sum_{i=1}^{n}\left(\alpha_i Q_i, P_{\text{pub}}\right), e\left(\sum_{i=1}^{n}(W_i + \beta_i PK_i), Z\right)\right). \tag{2}$$

(4) If equation (2) holds, it emits 1 and the verifier accepts the aggregate signature σ; otherwise, it emits 0 and the verifier rejects σ.

Our proposed CLAS scheme is correct if and only if the single signature and aggregate signature generated using our scheme can satisfy equations (1) and (2), respectively, where the correctness of the scheme is elaborated as follows:

$$\begin{aligned}
e(V_i, P) &= e(\alpha_i D_i + (w_i + \beta_i x_i)Z, P) \\
&= e(\alpha_i s Q_i, P) e((w_i + \beta_i x_i)Z, P) \\
&= e(\alpha_i Q_i, sP) e((w_i + \beta_i x_i)P, Z) \\
&= e\left(\alpha_i Q_i, P_{\text{pub}}\right) e(w_i P + \beta_i x_i P, Z) \\
&= e\left(\alpha_i Q_i, P_{\text{pub}}\right) e(W_i + \beta_i PK_i, Z), \\
e(V, P) &= e\left(\sum_{i=1}^{n}(\alpha_i D_i + (w_i + \beta_i x_i)Z), P\right) \\
&= e\left(\sum_{i=1}^{n}\alpha_i s Q_i, P\right) e\left(\sum_{i=1}^{n}(w_i + \beta_i x_i)Z, P\right) \\
&= e\left(\sum_{i=1}^{n}\alpha_i Q_i, sP\right) e\left(\sum_{i=1}^{n}(w_i + \beta_i x_i)P, Z\right) \\
&= e\left(\sum_{i=1}^{n}\alpha_i Q_i, P_{\text{pub}}\right) e\left(\sum_{i=1}^{n}(W_i + \beta_i PK_i), Z\right).
\end{aligned} \tag{3}$$

5. Security Analysis

In this section, we analyze the security of our proposed CLAS scheme. We first give the security model of a CLAS scheme and then prove that our proposal can satisfy signature unforgeability under the security model. At last, we demonstrate a comparative summary of the security between our CLAS scheme and three recently published CLAS schemes.

5.1. Security Model. There exist two types of adversaries in the CLAS security model: $\mathcal{A}_I$ and $\mathcal{A}_{II}$. $\mathcal{A}_I$ simulates an outside attacker, who cannot obtain the system master key but can replace any user's public key. $\mathcal{A}_{II}$ simulates a KGC,

an internal attacker, who can obtain the system master key but cannot replace any user's public key.

Definition 1. The security model of a CLAS scheme is defined by two games (denoted by Game1 and Game2) played between an adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$ and a challenger $\mathcal{C}$; more details are defined below.

$\mathcal{A}$ can access the following six random oracle machines in the security model.

5.1.1. Setup. $\mathcal{C}$ executes the Setup algorithm to generate the system master key s and params. For different types of adversaries, $\mathcal{C}$ will make a corresponding response.

5.1.2. Reveal-Partial-Key. When the challenger $\mathcal{C}$ receives a partial private key query from $\mathcal{A}$ for a user with the identity ID_i, $\mathcal{C}$ first checks if $ID_i = ID_{tu}$ holds. If it holds, it aborts; otherwise, it checks if there is a record corresponding to the identity ID_i in the list L_D. If it exists, then D_i is sent to $\mathcal{A}$; otherwise, it generates D_i, sends it to $\mathcal{A}$, and stores it in the list L_D.

5.1.3. Reveal-Secret-Key. When the challenger $\mathcal{C}$ receives a secret value query from $\mathcal{A}$ for a user with the identity ID_i, $\mathcal{C}$ first checks if $ID_i = ID_{tu}$ holds. If it holds, it aborts; otherwise, it checks if there is a record corresponding to the identity ID_i in the list L_x. If it exists, then x_i is sent to $\mathcal{A}$; otherwise, it generates x_i, sends it to $\mathcal{A}$, and stores it in the list L_x.

5.1.4. Reveal-Public-Key. When the challenger $\mathcal{C}$ receives a public key query from $\mathcal{A}$ for a user with the identity ID_i, $\mathcal{C}$ first checks if there is a record corresponding to the identity ID_i in the list L_{PK}. If it exists, then PK_i is sent to $\mathcal{A}$; otherwise, it generates PK_i, sends it to $\mathcal{A}$, and stores it in the list L_{PK}.

5.1.5. Replace-Public-Key. When the challenger $\mathcal{C}$ receives a query that replaces the public key on the identity ID_i with $A's$ choice of public key PK_i^*, $\mathcal{C}$ first checks if there is a record corresponding to the identity ID_i in the list L_{PK}. If it exists, then it updates the corresponding item (ID_i, x_i, PK_i, D_i) to (ID_i, x_i, PK_i^*, D_i) in the list L_{PK}; otherwise, it aborts.

5.1.6. Sign. When the challenger $\mathcal{C}$ receives a signature query on the message m_i with the signer's identity ID_i, $\mathcal{C}$ first checks whether the target user ID_i has been created. If the user ID_i has not been created, it aborts; otherwise, if the target user ID_i has been created and the related user public key PK_i has not been replaced, then a valid signature σ_i is returned; otherwise, if the target user ID_i has been created and the corresponding user public key PK_i has been replaced with PK_i^*, then a signature σ_i^* is returned.

We next define two games to describe two different types of attackers in the CLAS scheme.

(1) Game1. The challenger $\mathcal{C}$ interacts with the adversary $\mathcal{A}_I$ as follows:

(1) $\mathcal{C}$ inputs a security parameter k and generates the system master key s and the system parameter list params by running the Setup algorithm. Then, $\mathcal{C}$ sends params to $\mathcal{A}_I$ and keeps s secret.

(2) $\mathcal{A}_i$ can access any hash oracle and Reveal-Partial-Key, Reveal-Secret-Key, Reveal-Public-Key, Replace-Public-Key, and Sign queries at any phase.

(2) Forgery. $\mathcal{A}_I$ outputs an aggregate signature σ^* with respect to n pairs $(ID_i^*, m_i^*, PK_i^*, \sigma_i^*)$, where $1 \le i \le n$. We say that $\mathcal{A}_I$ wins Game1 if and only if the following conditions are met:

(1) σ^* is a valid aggregate signature with respect to pairs $(ID_i^*, m_i^*, PK_i^*, \sigma_i^*)$, where $1 \le i \le n$.

(2) The targeted identity ID_i^* has not been submitted during the Reveal-Partial-Key query.

(3) (ID_i^*, m_i^*) has not been submitted during the Sign query.

(3) Game2. The challenger $\mathcal{C}$ interacts with the adversary $\mathcal{A}_{II}$ as follows:

(1) $\mathcal{C}$ inputs a security parameter k and generates the system master key s and the system parameter list params by running the Setup algorithm. Then, $\mathcal{C}$ returns params and s to $\mathcal{A}_{II}$.

(2) $\mathcal{A}_{II}$ can access any hash oracle and Reveal-Partial-Key, Reveal-Public-Key, and Sign queries at any phase.

(4) Forgery. $\mathcal{A}_{II}$ outputs an aggregate signature σ^* with respect to pairs $(ID_i^*, m_i^*, PK_i^*, \sigma_i^*)$, where $1 \le i \le n$. We say that $\mathcal{A}_{II}$ wins Game2 if and only if the following conditions are met:

(1) σ^* is a valid aggregate signature with respect to user-message-public key-signature pairs $(ID_i^*, m_i^*, PK_i^*, \sigma_i^*)$, where $1 \le i \le n$.

(2) The targeted identity ID_i^* has not been submitted during the Reveal-Secret-Key query.

(3) (ID_i^*, m_i^*) has not been submitted during the Sign query.

5.2. Security Proof. In the section, we will prove that our proposed CLAS scheme is secure under the security model presented in Section 5.1. Our security proof consists of the following two parts: (1) the CLAS scheme is unforgeable to type 1 adversary $\mathcal{A}_I$ and (2) the CLAS scheme is unforgeable to type 2 adversary $\mathcal{A}_{II}$.

Theorem 1. *Our proposed CLAS scheme is existentially unforgeable against the adversary $\mathscr{A}_I$, if the CDH problem is difficult to solve in G_1.*

Proof. We can prove the unforgeability of our CLAS scheme against $\mathscr{A}_i$ with Game1 that involves an adversary $\mathscr{A}_I$ and a simulator C.

Given a random instance of the CDH problem $(P, Q_1 = aP, Q_2 = bP)$, where P is a generator of G_1, our goal is to calculate the value of abP by solving the CDH problem. The specific proof process is as follows:

(i) Setup: C randomly selects ID_{tu} as the identity of the target user challenged, sets $P_{\text{pub}} = Q_1 = aP$, and generates and sends the system parameter $\text{params} = \{G_1, G_2, P, e, q, P_{\text{pub}}, H_1, H_2, h_1, h_2\}$ to $\mathscr{A}_I$. $\mathscr{A}_I$ executes the following queries.

(ii) H_1 query: C maintains a list L_{H_1} whose structure is $(ID_i, \eta_i, \zeta_i, Q_i)$, and all the elements in L_{H_1} are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_i$, it first checks whether the tuple $(ID_i, \eta_i, \zeta_i, Q_i)$ exists in L_{H_1}; if it exists, it sends Q_i to $\mathscr{A}_I$; otherwise, C randomly selects $\zeta_i \in \{0, 1\}$ and $\eta_i \in Z_q^*$. If $\zeta_i = 0$, $Q_i = \eta_i P$; otherwise, if $\zeta_i = 1$, $Q_i = \eta_i Q_2 = \eta_i bP$. It sends Q_i to $\mathscr{A}_i$ and stores $(ID_i, \eta_i, \zeta_i, Q_i)$ to L_{H_1}.

(iii) H_2 query: C maintains a list L_{H_2} whose structure is $(P_{\text{pub}}, \vartheta, Z)$, and all the elements in L_{H_2} are initialized to null. When C receives a query with P_{pub} from $\mathscr{A}_I$, it checks if a tuple $(P_{\text{pub}}, \vartheta, Z)$ exists in L_{H_2}; if it exists, it sends Z to $\mathscr{A}_I$; otherwise, C randomly selects $\vartheta \in Z_q^*$ and computes $Z = \vartheta P$. It sends Z to $\mathscr{A}_i$ and stores $(P_{\text{pub}}, \vartheta, Z)$ to L_{H_2}.

(iv) h_1 query: C maintains a list L_{h_1} whose structure is $(ID_i, PK_i, W_i, \alpha_i)$, and all the elements in L_{h_1} are initialized to null. When C receives a query with the tuple (ID_i, PK_i, W_i) from $\mathscr{A}_I$, it checks whether a tuple $(ID_i, PK_i, W_i, \alpha_i)$ exists in L_{h_1}; if it exists, it sends α_i to $\mathscr{A}_I$; otherwise, C randomly selects α_i. It returns α_i to $\mathscr{A}_I$ and stores $(ID_i, PK_i, W_i, \alpha_i)$ to L_{h_1}.

(v) h_2 query: C maintains a list L_{h_2} whose structure is $(m_i, ID_i, PK_i, W_i, \beta_i)$, and all the elements in L_{h_2} are initialized to null. When C receives a query with the tuple (m_i, ID_i, PK_i, W_i) from $\mathscr{A}_I$, it checks if a tuple $(m_i, ID_i, PK_i, W_i, \beta_i)$ exists in L_{h_2}; if it exists, it sends β_i to $\mathscr{A}_I$; otherwise, C randomly selects β_i. It returns β_i to $\mathscr{A}_I$ and stores $(m_i, ID_i, PK_i, W_i, \beta_i)$ to L_{h_2}.

(vi) Reveal-Partial-Key queries: C maintains a list L_D whose structure is (ID_i, D_i), and all the elements in L_D are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_I$, it first checks whether $ID_i = ID_{tu}$; if it holds, it outputs $\perp$; otherwise, it checks if a tuple (ID_i, D_i) exists in L_D; if it exists, it sends D_i to $\mathscr{A}_I$; otherwise, C recalls the corresponding tuple $(ID_i, \eta_i, \zeta_i, Q_i)$ from the list L_{H_1} and computes $D_i = \eta_i P_{\text{pub}} = a\eta_i P$. It sends D_i to $\mathscr{A}_I$ and stores (ID_i, D_i) to L_D.

(vii) Reveal-Secret-Key queries: C maintains a list L_x whose structure is (ID_i, x_i), and all the elements in L_x are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_I$, it first checks whether $ID_i = ID_{tu}$; if it holds, it outputs $\perp$; otherwise, it checks if a tuple (ID_i, x_i) exists in L_x; if it exists, it sends x_i to $\mathscr{A}_I$; otherwise, C randomly selects x_i. It sends x_i to $\mathscr{A}_I$ and stores x_i to (ID_i, x_i).

(viii) Reveal-Public-Key queries: C maintains a list L_{PK} whose structure is (ID_i, PK_i), and all the elements in L_{PK} are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_I$, it first checks whether a tuple (ID_i, PK_i) exists in L_{PK}; if it exists, it sends PK_i to $\mathscr{A}_I$; otherwise, C accesses L_x to get x_i and computes $PK_i = x_i P$. It sends PK_i to $\mathscr{A}_I$ and stores (ID_i, PK_i) to L_{PK}.

(ix) Replace-Public-Key queries: when C receives a query with the tuple (ID_i, PK_i^*) from $\mathscr{A}_I$, in response, C replaces the real public key PK_i of ID_i with PK_i^* chosen by $\mathscr{A}_I$ in L_{PK}.

(x) Sign queries: when C receives a query with the tuple (m_i, ID_i, PK_i) from $\mathscr{A}_I$, C accesses L_{H_1}, L_{H_2}, L_{h_1}, and L_{h_2} to get ζ_i, Q_i, Z, α_i, and β_i respectively. Furthermore, C chooses a random $w_i \in Z_q^*$ and computes $W_i = w_i P$; if $\zeta_i = 0$, C computes $V_i = \eta_i \alpha_i P + (w_i + \beta_i x_i)\vartheta P$; otherwise, if $\zeta_i = 1$, C computes $V_i = \eta_i \alpha_i bP + (w_i + \beta_i x_i)\vartheta P$. C sends $\sigma_i = (W_i, V_i)$ to $\mathscr{A}_I$ as the signature on the message m_i with the identity ID_i and the public key PK_i.

(xi) Forgery: finally, $\mathscr{A}_I$ outputs a forged aggregate signature $\sigma^* = (W^*, V^*)$ from message-identity-public key pairs (m_i^*, ID_i^*, PK_i^*), where $1 \le i \le n$. If all $\zeta_i = 0$ holds, $\mathscr{A}_I$ aborts; otherwise, without loss of generality, let $ID_{tu} = ID_1$, that is, $\zeta_1 = 1$, $\zeta_i = 0$, $(2 \le i \le n)$, and then the forged signature can make the following equation hold:

$$e(V^*, P) = e\left(\sum_{i=1}^{n} \alpha_i^* Q_i^*, P_{pub}\right) e\left(\sum_{i=1}^{n} (W_i^* + \beta_i^* PK_i^*), Z\right), \tag{4}$$

where $Q_i^* = \eta_i^* P$ $(2 \le i \le n)$, $Q_1^* = \eta_1^* bP$, $Z = \vartheta P$, $V^* = \sum_{i=1}^{n} V_i^*$, and $W^* = \{W_1^*, W_2^*, \ldots, W_n^*\}$.

Furthermore, the derivation process is shown as

$$
\begin{aligned}
e(V^*, P) &= e\left(\sum_{i=1}^{n}(\alpha_i^* D_i^* + (w_i^* + \beta_i^* x_i^*)Z), P\right) \\
\Longrightarrow e(V^*, P) &= e(\alpha_1^* aQ_1^*, P)e\left(\sum_{i=2}^{n}\alpha_i^* aQ_i^*, P\right)e\left(\sum_{i=1}^{n}(w_i^* + \beta_i^* x_i^*)Z, P\right) \\
\Longrightarrow e(\alpha_1^* aQ_1^*, P) &= e(V^*, P)\left[e\left(\sum_{i=2}^{n}\alpha_i^* aQ_i^*, P\right)e\left(\sum_{i=1}^{n}(w_i^* + \beta_i^* x_i^*)Z, P\right)\right]^{-1} \\
\Longrightarrow e(\alpha_1^* \eta_1^* abP, P) &= e(V^*, P)\left[e\left(\sum_{i=2}^{n}\alpha_i^* aQ_i^*, P\right)e\left(\sum_{i=1}^{n}(w_i^* + \beta_i^* x_i^*)Z, P\right)\right]^{-1} \\
\Longrightarrow \alpha_1^* \eta_1^* abP &= V^* - \sum_{i=2}^{n}\alpha_i^* aQ_i^* - \sum_{i=1}^{n}(w_i^* + \beta_i^* x_i^*)Z \\
\Longrightarrow abP &= \left(V^* - \sum_{i=2}^{n}\alpha_i^* aQ_i^* - \sum_{i=1}^{n}(w_i^* + \beta_i^* x_i^*)Z\right)(\alpha_1^* \eta_1^*)^{-1}.
\end{aligned} \tag{5}
$$

However, this is in contradiction with the CDH assumption, so the single signature and the aggregate signature generated by our proposed scheme satisfy the unforgeability. □

Theorem 2. *Our proposed CLAS scheme is existentially unforgeable against the adversary $\mathscr{A}_{II}$, if the CDH problem is difficult to solve in G_1.*

Proof. We can prove the unforgeability of our CLAS scheme against $\mathscr{A}_{II}$ with Game2 that involves an adversary $\mathscr{A}_{II}$ and a simulator C.

Given a random instance of the CDH problem $(P, Q_1 = aP, Q_2 = bP)$, where P is the generator of G_1, our goal is to calculate the value of abP by solving the CDH problem. The specific proof process is as follows:

(i) Setup: C randomly selects ID_{tu} as the identity of the target user challenged, sets $P_{\text{pub}} = \lambda P$, and generates and sends params $= \{G_1, G_2, P, e, q, P_{\text{pub}}, H_1, H_2, h_1, h_2\}$ and system master key λ to $\mathscr{A}_{II}$. $\mathscr{A}_{II}$ executes the following queries: h_1, h_2, and *Reveal-Secret-Value queries* are the same as the corresponding queries in Theorem 1 Since $\mathscr{A}_{II}$ can access the system master key, there is no need for the *Reveal-Partial-Key queries* and *Replace-Public-Key queries*.

(ii) H_1 query: C maintains a list L_{H_1} whose structure is (ID_i, η_i, Q_i), and all the elements in L_{H_1} are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_{II}$, it first checks whether the tuple (ID_i, η_i, Q_i) exists in L_{H_1}; if it exists, it sends Q_i to $\mathscr{A}_{II}$; otherwise, C randomly selects $\eta_i \in Z_q^*$, sets $Q_i = \eta_i P$, sends Q_i to $\mathscr{A}_{II}$, and stores (ID_i, η_i, Q_i) to L_{H_1}.

(iii) H_2 query: C maintains a list L_{H_2} whose structure is $(P_{\text{pub}}, \vartheta, Z)$, and all the elements in L_{H_2} are initialized to null. When C receives a query with P_{pub} from $\mathscr{A}_{II}$, it checks if a tuple $(P_{\text{pub}}, \vartheta, Z)$ exists in L_{H_2}; if it exists, it sends Z to $\mathscr{A}_{II}$; otherwise, C randomly selects $\vartheta \in Z_q^*$ and computes $Z = \vartheta Q_1 = \vartheta aP$. It sends Z to $\mathscr{A}_{II}$ and stores (P_{pub}, ϑ, Z) to L_{H_2}.

(iv) Reveal-Public-Key queries: C maintains a list L_{PK} whose structure is (ID_i, ζ_i, PK_i), and all the elements in L_{PK} are initialized to null. When C receives a query with the identity ID_i from $\mathscr{A}_{II}$, it first checks whether a tuple (ID_i, ζ_i, PK_i) exists in L_{PK}; if it exists, it sends PK_i to $\mathscr{A}_{II}$; otherwise, C selects a random value $\zeta_i \in \{0, 1\}$; if $\zeta_i = 0$, C accesses L_x to get x_i and computes $PK_i = x_i P$; otherwise, if $\zeta_i = 1$, C randomly chooses $x_i \in Z_q^*$ and computes $PK_i = x_i Q_2 = x_i bP$. It sends PK_i to $\mathscr{A}_{II}$ and stores (ID_i, ζ_i, PK_i) to L_{PK}.

(v) Sign queries: when C receives a query with the tuple (m_i, ID_i, PK_i) from $\mathscr{A}_{II}$, C accesses L_{H_1}, L_{H_2}, L_{h_1}, and L_{h_2} to get Q_i, ζ_i, Z, α_i, and β_i, respectively. Furthermore, C chooses a random $w_i \in Z_q^*$ and computes $W_i = w_i P$; if $\zeta_i = 0$, C computes $V_i = \lambda \alpha_i Q_i + (w_i + \beta_i x_i)Z$; otherwise, if $\zeta_i = 1$, C computes $V_i = \lambda \alpha_i Q_i + (w_i + b\beta_i x_i)Z$. It sends $\sigma_i = (W_i, V_i)$ to $\mathscr{A}_{II}$ as the signature on the message m_i with the identity ID_i and the public key PK_i.

(vi) Forgery: finally, $\mathscr{A}_{ii}$ outputs a forged aggregate signature $\sigma^* = (W^*, V^*)$ from message-identity-public key pairs (m_i^*, ID_i^*, PK_i^*), where $1 \le i \le n$. If all $\zeta_i = 0$ holds, $\mathscr{A}_{II}$ aborts; otherwise, without loss of generality, let $ID_{tu} = ID_1$, that is, $\zeta_1 = 1$, $\zeta_i = 0$ $(2 \le i \le n)$, and then the forged signature can make the following equation hold:

$$
e(V^*, P) = e\left(\sum_{i=1}^{n}\alpha_i^* Q_i^*, P_{pub}\right)e\left(\sum_{i=1}^{n}(W_i^* + \beta_i^* PK_i^*), Z\right), \tag{6}
$$

where $Q_i^* = \eta_i^* P$, $PK_1^* = x_1^* bP$, $PK_i^* = x_i^* P$ $(2 \le i \le n)$, $Z = \vartheta aP$, $V^* = \sum_{i=1}^{n} V_i^*$, and $W^* = \{W_1^*, W_2^*, \ldots, W_n^*\}$.

Furthermore, the derivation process is shown as

$$
\begin{aligned}
& e(V^*, P) = e\left(\sum_{i=1}^{n}(\alpha_i^* D_i^* + (w_i^* + \beta_i^* x_i^*)Z), P\right) \\
\Longrightarrow\ & e(V^*, P) = e\left(\sum_{i=1}^{n}\lambda\alpha_i^* Q_i^*, P\right) e(\beta_1^* PK_1^*, Z) e\left(\sum_{i=1}^{n} W_i^* + \sum_{i=2}^{n}\beta_i^* PK_i^*, Z\right) \\
\Longrightarrow\ & e(\beta_1^* PK_1^*, Z) = e(V^*, P)\left[e\left(\sum_{i=1}^{n}\lambda\alpha_i^* Q_i^*, P\right) e\left(\vartheta a\left(\sum_{i=1}^{n} W_i^* + \sum_{i=2}^{n}\beta_i^* PK_i^*\right), P\right)\right]^{-1} \\
\Longrightarrow\ & \beta_1^* x_1^* \vartheta abP = V^* - \sum_{i=1}^{n}\lambda\alpha_i^* Q_i^* - \vartheta a\left(\sum_{i=1}^{n} W_i^* + \sum_{i=2}^{n}\beta_i^* PK_i^*\right) \\
\Longrightarrow\ & abP = (\beta_1^* x_1^* \vartheta)^{-1}\left(V^* - \sum_{i=1}^{n}\lambda\alpha_i^* Q_i^* - \vartheta a\left(\sum_{i=1}^{n} W_i^* + \sum_{i=2}^{n}\beta_i^* PK_i^*\right)\right).
\end{aligned}
\tag{7}
$$

However, this is in contradiction with the CDH assumption, so the single signature and the aggregate signature generated by our proposed scheme satisfy the unforgeability. □

6. Security Comparisons and Performance Analysis

In this section, we first compare the security of the new CLAS scheme and the other three CLAS schemes and then further analyze the performance advantages of the new CLAS scheme by evaluating the computation overhead.

6.1. Security Comparisons. In the section, we compare the security of our proposed CLAS scheme with that of the other three CLAS schemes [21, 25, 29]. For ease of description, let $\mathscr{A}_I$ and $\mathscr{A}_{II}$ denote the type 1 and the type 2 adversaries, respectively. Furthermore, the two types of adversaries are divided into three levels [30], where B_{i1} denotes general adversary, B_{i2} denotes strong adversary, and B_{i3} denotes superadversary, respectively, and $i \in \{1, 2\}$; the value of i corresponds to the type i adversary. √ denotes it can satisfy the corresponding security requirement, and × denotes it cannot satisfy the corresponding security requirement. W denotes the weaker security, and S denotes the stronger security under the corresponding attack types. *SP* denotes the security performance. The security comparison of various schemes is shown in Table 1.

From Table 1, we can find that the first two schemes (i.e., Gong et al.'s scheme [21] and Liu et al.'s scheme [25]) cannot meet all security attributes. Especially for Gong et al.'s two CLAS schemes [21], under the attacks of $\mathscr{A}_i$ and $\mathscr{A}_{ii}$ adversaries, none of them could satisfy the security level of B_3. In contrast, Li's CLAS scheme and our proposed CLAS scheme can meet all the security requirements.

6.2. Performance Analysis. In this section, we performed a performance analysis of the newly proposed CLAS scheme by comparing the computation overhead of our scheme with that of Li et al.'s scheme. To achieve a credible security level, we select q and p as 160-bit and 512-bit prime numbers, respectively. An Ate pairing e: $G_1 \times G_1 \longrightarrow G_2$ is used in our experiments, where G_1 and G_2 are two cyclic groups with the same order q, defined on the supersingular elliptic curve $E(F_p)$: $y^2 = x^3 + 1$.

We have implemented Li et al.'s scheme and our new scheme with the MIRACL library [31] on a Lenovo computer with Windows 7 operating system. And its hardware configuration is Intel I5-3470 3.20 GHz CPU and 4G bytes of memory. For the sake of simplicity, we firstly define the corresponding relation-related symbol-operation-execution time, as shown in Table 2.

Because Setup, *Partial-Key-Gen*, and *Private-Key-Gen* phases are executed by the PKG or user, and all of them are one-time operation, we focus on the analysis and comparison of computational costs in Sign, Verify, Aggregate, and Aggregate-Verify phases. Since the addition and multiplication of numbers in Z_q^* will only generate less computational overhead, we can ignore them.

In the *Sign* phase, the user in Li et al.'s scheme needs to perform two general hash operations in Z_q^*, one map-to-point hash operation in G_1, three point addition operations in G_1, and five point multiplication operations in G_1. Therefore, the time for generating a signature in the Sign phase is $2T_{m-s} + T_{m-p} + 3T_{ecc-pa} + 5T_{ecc-pm}$, whereas the user in our proposal needs to perform two general hash operations in Z_q^*, one map-to-point hash operation in G_1, two point addition operations in G_1, and three point multiplication operations in G_1. Therefore, the time for generating a signature in the Sign phase is $2T_{m-s} + T_{m-p} + 2T_{ecc-pa} + 3T_{ecc-pm}$ milliseconds.

In the Verify phase, the verifier in Li et al.'s scheme needs to execute two general hash operations in Z_q^*, two map-to-point hash operations in G_1, two point addition operations

Table 1: Security comparisons.

	B_{11}	B_{12}	B_{13}	SP	B_{21}	B_{22}	B_{23}	SP
Gong et al.'s scheme [21]	×	√	×	S	√	×	×	W
Liu et al.'s scheme [25]	√	√	√	S	×	×	×	W
Li et al.'s scheme [29]	√	√	√	S	√	√	√	S
Our proposed scheme	√	√	√	S	√	√	√	S

Table 2: Symbol-operation-execution time.

Symbol	Operation	Time (ms)
T_{m-s}	The time of executing a general hash operation in Z_q^*	0.0002
T_{m-p}	The time of executing a map-to-point operation in G_1	9.773
T_{ecc-pa}	The time of executing a point addition operation in G_1	0.022
T_{ecc-pm}	The time of executing a point multiplication operation in G_1	3.740
T_{bp}	The time of executing a bilinear pairing operation	11.515

in G_1, two point multiplication operations in G_1, and three bilinear pairing operations. Therefore, the time for verifying a signature in the Verify phase is $2T_{m-s} + 2T_{m-p} + 2T_{ecc-pa} + 2T_{ecc-pm} + 3T_{bp}$, whereas the verifier in our proposal needs to perform two general hash operations in Z_q^*, two map-to-point hash operations in G_1, one point addition operation in G_1, two point multiplication operations in G_1, and three bilinear pairing operations. Therefore, the time for verifying a signature in the Verify phase is $2T_{m-s} + 2T_{m-p} + T_{ecc-pa} + 2T_{ecc-pm} + 3T_{bp}$ milliseconds.

In the *Aggregate* phase, the aggregator in Li et al.'s scheme needs to execute $n-1$ point addition operations in G_1, whereas the aggregator in our proposal needs to execute $n-1$ point addition operations in G_1. We can find that the running time of the Aggregate phase in the two schemes is equal to $(n-1)T_{ecc-pa}$ milliseconds.

In the Aggregate – Verify phase, the aggregate verifier in Li et al.'s scheme needs to execute $2n$ general hash operations in Z_q^*, $n+1$ map-to-point hash operations in G_1, $4n-2$ point addition operations in G_1, $2n$ point multiplication operations in G_1, and three bilinear pairing operations. Therefore, the running time of the Aggregate – Verify phase is $2nT_{m-s} + (n+1)T_{m-p} + (4n-2)T_{ecc-pa} + 2nT_{ecc-pm} + 3T_{bp}$ milliseconds, whereas the verifier in our proposal needs to perform $2n$ general hash operations in Z_q^*, $n+1$ map-to-point hash operations in G_1, $3n-2$ point addition operations in G_1, $2n$ point multiplication operations in G_1, and three bilinear pairing operations. Therefore, the running time of the Aggregate – Verify phase is $2nT_{m-s} + (n+1)T_{m-p} + (3n-2)T_{ecc-pa} + 2nT_{ecc-pm} + 3T_{bp}$ milliseconds.

Suppose that $n = 100$, that is, there are 100 signatures that need to be generated, verified, aggregated, and aggregate-verified. From the results in Figure 3, we can see that Li et al.'s scheme has the same computation overhead as our new CLAS scheme in the Aggregate phase, whereas in Sign, Verify, and Aggregate – Verify phases, the computation cost of our scheme is lower than that of Li et al.'s scheme. Especially in the Sign phase, the computation cost of our scheme is reduced by 26 percentage points compared with that of Li et al.'s scheme. In summary, our presented CLAS scheme reduces the computation cost while meeting security requirements.

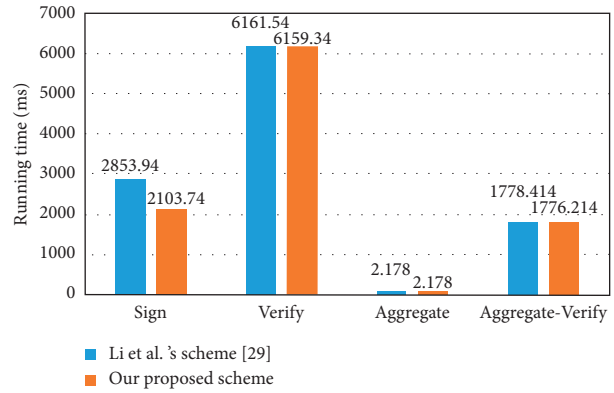

Figure 3: Computation cost comparisons.

7. Conclusion

Digital signature can provide secure routing authentication, privacy protection, integrity, and nonrepudiation. To solve the above problems, several CLAS schemes have been introduced recently. Unfortunately, most existing CLAS schemes have been found to have security flaws or have unsatisfactory performance in computation and communication costs. To avoid the above issues and better fix the problem of secure routing authentication in resource-constrained VANETs, we put forward a new CLAS scheme. The security analysis demonstrates that the new CLAS scheme is provably secure and is able to satisfy the security attributes in VANETs. The specific performance evaluation shows that the new CLAS scheme can achieve a novel security level while reducing the computation cost. Our CLAS scheme is robust against all types of attacks, which makes it more suitable for performing secure routing in resource-constrained VANETs.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61902115, 61972294, and 61932016) and the Opening Project of Guangdong Provincial Key Laboratory of Data Security and Privacy Protection (No. 2017B030301004-11).

References

[1] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 4, pp. 386–399, 2006.

[2] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Proceedings of the Australasian Conference on Information Security and Privacy*, Springer, Sydney, pp. 200–211, Australia, July 2004.

[3] J. A. Misener, "Vehicle-infrastructure integration (vii) and safety: rubber and radio meets the road in California," *Intellimotion*, vol. 11, no. 2, pp. 1–3, 2005.

[4] E. S. Ismail, N. M. F. Tahat, and R. R. Ahmad, "A new digital signature scheme based on factoring and discrete logarithms," *Journal of Mathematics and Statistics*, vol. 4, no. 4, pp. 222–225, 2008.

[5] Q. Lin, J. Li, Z. Huang, W. Chen, and J. Shen, "A short linearly homomorphic proxy signature scheme," *IEEE Access*, vol. 6, pp. 12966–12972, 2018.

[6] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *Journal of Computer Security*, vol. 15, no. 1, pp. 39–68, 2007.

[7] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen, "ECPP: efficient conditional privacy preservation protocol for secure vehicular communications," in *Proceedings of the 27th Conference on Computer Communications, INFOCOM 2008*, pp. 1229–1237, IEEE, Phoenix, AZ, USA, April 2008.

[8] Y. Sun, R. Lu, X. Lin, X. Shen, and J. Su, "An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 7, pp. 3589–3603, 2010.

[9] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the 1984 Workshop on the Theory and Application of Cryptographic Techniques*, pp. 47–53, Springer, Santa Barbara, CA, USA, August 1984.

[10] K.-A. Shim, "An ID-based aggregate signature scheme with constant pairing computations," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1873–1880, 2010.

[11] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 425–437, 2015.

[12] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[13] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, Springer, Taipei, Taiwan, November 2003.

[14] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, p. 1, 2018.

[15] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "On the security of certificateless signature schemes from Asiacrypt 2003," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 13–25, Springer, Xiamen, China, December 2005.

[16] B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, "Key replacement attack against a generic construction of certificateless signature," in *Proceedings of the 2006 Australasian Conference on Information Security and Privacy*, pp. 235–246, Springer, Melbourne, Australia, July 2006.

[17] M. H. Au, Y. Mu, J. Chen, D. S. Wong, J. K. Liu, and G. Yang, "Malicious KGC attacks in certificateless cryptography," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pp. 302–311, ACM, Singapore, March 2007.

[18] X. Li, K. Chen, and L. Sun, "Certificateless signature and proxy signature schemes from bilinear pairings," *Lithuanian Mathematical Journal*, vol. 45, no. 1, pp. 76–83, 2005.

[19] J. K. Liu, M. H. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pp. 273–283, ACM, Singapore, March 2007.

[20] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 416–432, Springer, Warsaw, Poland, May 2003.

[21] Z. Gong, Y. Long, X. Hong, and K. Chen, "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, 2007. SNPD 2007*, vol. 3, pp. 188–193, IEEE, Qingdao, China, July 2007.

[22] L. Zhang and F. Zhang, "A new certificateless aggregate signature scheme," *Computer Communications*, vol. 32, no. 6, pp. 1079–1085, 2009.

[23] H. Xiong, Z. Guan, Z. Chen, and F. Li, "An efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 219, pp. 225–235, 2013.

[24] D. He, M. Tian, and J. Chen, "Insecurity of an efficient certificateless aggregate signature with constant pairing computations," *Information Sciences*, vol. 268, pp. 458–462, 2014.

[25] H. Liu, S. Wang, M. Liang, and Y. Chen, "New construction of efficient certificateless aggregate signatures," *International Journal of Security and Its Applications*, vol. 8, no. 1, pp. 411–422, 2014.

[26] L. Cheng, Q. Wen, Z. Jin, H. Zhang, and L. Zhou, "Cryptanalysis and improvement of a certificateless aggregate signature scheme," *Information Sciences*, vol. 295, pp. 337–346, 2015.

[27] F. Zhang, L. Shen, and G. Wu, "Notes on the security of certificateless aggregate signature schemes," *Information Sciences*, vol. 287, pp. 32–37, 2014.

[28] S.-J. Horng, S.-F. Tzeng, P.-H. Huang, X. Wang, T. Li, and M. K. Khan, "An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks," *Information Sciences*, vol. 317, pp. 48–66, 2015.

[29] J. Li, H. Yuan, and Y. Zhang, "Cryptanalysis and improvement for certificateless aggregate signature," *Fundamenta Informaticae*, vol. 157, no. 1-2, pp. 111–123, 2018.

[30] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[31] M. Scott, *Miracla Multiprecision Integer and Rational Arithmetic C/C++ Library*, Shamus Software Ltd., Dublin, Ireland, 2003, http://indigo.ie/mscott.