Research Article

Certificateless-Based Anonymous Authentication and Aggregate Signature Scheme for Vehicular Ad Hoc Networks

Xin Ye,1 Gencheng Xu,2 Xueli Cheng,2 Yuedi Li,1 and Zhiguang Qin1

1School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, 610054, China
2School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan Province, 611731, China

Correspondence should be addressed to Gencheng Xu; xugencheng@std.uestc.edu.cn

Received 11 December 2020; Revised 18 January 2021; Accepted 11 February 2021; Published 16 March 2021

Academic Editor: Zhuojun Duan

Copyright © 2021 Xin Ye et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Development of Internet of Vehicles (IoV) has aroused extensive attention in recent years. The IoV requires an efficient communication mode when the application scenarios are complicated. To reduce the verifying time and cut the length of signature, certificateless aggregate signature (CL-AS) is used to achieve improved performance in resource-constrained environments like vehicular ad hoc networks (VANETs), which is able to make it effective in environments constrained by bandwidth and storage. However, in the real application scenarios, messages should be kept untamed, unleashed, and authentic. In addition, most of the proposed schemes tend to be easy to attack by signers or malicious entities which can be called coalition attack. In this paper, we present an improved certificateless-based authentication and aggregate signature scheme, which can properly solve the coalition attack. Moreover, the proposed scheme not only uses pseudonyms in communications to prevent vehicles from revealing their identity but also achieves considerable efficiency compared with state-of-the-art work, certificateless signature (CLS), and CL-AS schemes. Furthermore, it demonstrates that when focused on the existential forgery on adaptive chosen message attack and coalition attack, the proposed schemes can be proved secure. Also, we show that our scheme exceeds existing certification schemes in both computing and communication costs.

1. Introduction

With the rapid development of communication technology, various vehicles with powerful smart devices can communicate with each other. Therefore, such a novel application has aroused extensive interest in the society. This kind of application is commonly referred to as vehicle ad hoc networks (VANETs), which can provide guarantee for the distance between vehicles and reduce the probability of vehicle collision accidents, help car drivers navigate in real time, and improve the efficiency of traffic operation by communicating with other vehicles and network systems [1].

Although VANETs have a lot of merits, it has a long way to achieve a wide application. One of the obstacles is that the privacy is violated. Without proper privacy protection, malicious adversaries can collect vehicle information, such as routes or status, to perform attacks. Fortunately, using pseudonyms in communications can avoid this problem. Then, the vehicle can communicate with each other or with roadside unit (RSU) using a pseudonym, and no one can obtain the true identity of the vehicle except for the trusted authority (TA). Even if the messages between the vehicles and the RSUs are collected by hackers, it will not reveal identity privacy. VANETs have other problems such as privacy issues and being vulnerable to attack.

Recently, some novel schemes and algorithms are proposed to solve these problems. Lin et al. [2] proposed a blockchain-based protocol to reduce the verification cost and storage cost for vehicles. Kumar et al. [3] proposed an efficient scheme using path signature to resist Sybil attack. Jiang et al. [4] proposed an anonymous authentication scheme (AAAS) in VANETs, which adopts group signature mechanism to provide more efficient anonymous authentication service for vehicles. Zheng et al. [5] demonstrated a
certificateless group signature anonymous authentication scheme for VANETs, which shortens the length of the signature and improves the efficiency of the signature. Among various schemes, we find that Kamil et al.’s scheme [6] has a significant efficiency. However, we find that the scheme cannot resist coalition attack which is launched by two collusive vehicles. For example, two vehicles can maliciously exchange their locations to generate their signatures which can be verified successfully so that they can hide their real locations which may lead to serious consequences. The detailed description and analysis are shown in Subsection 4.3. We make the RSU both the aggregator and the verifier and add a random list to properly solve the problem. Our main contributions in this paper are as follows:

(i) Prove that Kamil et al.’s schemes are not secure enough to defend against attacks from malicious vehicles and propose a solution to settle the problem

(ii) Propose an improved certificateless-based authentication and aggregate signature scheme in VANETs, and prove that the scheme can perfectly resist the coalition attacks and its correctness

(iii) Use the efficiency analysis and simulation to show the superiority of our scheme in efficiency and practicality

The rest of this paper is organized as follows. In Section 2, we discuss related works of CLS and CL-AS schemes in VANETs. In Section 3, we describe related concepts and models. In Section 4, we analyze Kamil et al.’s scheme and prove that the scheme cannot resist the coalition attack. We propose our proposed scheme in Section 5 in detail. Experiments and results analysis are described in Section 6. We conclude this paper in Section 7.

2. Related Works

To settle the problem of security and some privacy requirements in VANETs, a number of professors and scholars [7–9] proposed a kind of new scheme called Public Key Infrastructure-based (PKI-based) authentication schemes. In their schemes, they either tried to make vehicles compute more to verify the signatures from other vehicles or assume that there exists a trusted certificate authority to issue and maintain certificates of various vehicles. However, the assumption may be unrealistic because a single node cannot afford the oceans of calculation.

Later, a new kind of signature scheme called identity-based signature (IBS) scheme is widely discussed. For example, Liu et al. [10] proposed an IBS scheme which can take the user’s identity as the public key, and the private key is generated by public key generation PKG, which can reduce a single node’s burden. However, IBS has inherent problems about key escrow which is generated by user’s identity.

In Al-Riyami and Paterson’s scheme [11], they firstly introduce the certificateless public key cryptography. In recent years, a lot of researches on CLS and CL-AS schemes with bilinear pairing have been carried on by relevant researchers [12–14]. In their schemes, key generation center (KGC) uses its master key and the user’s identity information to calculate a part of the private key and send it to the user, whereafter the user combines part of the private key and his/her secret value together to generate the user’s real private key which can protect the user’s privacy and make the system secure. The above scheme uses the bilinear pairing which costs relatively large computation.

The elliptic curve cryptography is chosen to use in the CLS and CL-AS because of its high efficiency. In Xie et al.’s scheme [15], they proposed rigorous security proof that shows the scheme is able to resist various malicious attacks and ensure privacy protection. In the field of health care, Du et al. [16] proposed a CLAS scheme with high efficiency and low latency which can be more suitable to apply to the field of healthcare. In 2018, Cui et al. [17] demonstrated their novel CLS and CL-AS scheme with ECC, which significantly reduces computing time during sign and verification process. Kamil et al. [6] declared that the scheme proposed by Cui et al. is not secured against the signature forgery attack, and they advanced an improved signature scheme for VANETs. They claimed that their proposed scheme can address all the needs of VANETs about security and privacy. However, we will demonstrate and prove that their scheme cannot resist coalition attacks and our improved scheme can resist the attack and achieves a better performance.

3. Preliminaries

3.1. Elliptic Curve Cryptography. As widely used in the cryptographic, the elliptic curve cryptography is an excellent algorithm which has an extremely high efficiency and a relatively excellent security. It can use much fewer bits to encrypt messages of the same length than the RSA algorithm in the field of public key cryptography. Because of its fewer calculation parameters, shorter bond length, and less time cost, the elliptic curve cryptography can be perfectly applied to application scenarios of VANETs. We will give the following three definitions to describe the elliptic curve cryptography.

Definition 1 (Elliptic curve definition). Our scheme uses an elliptical encryption algorithm with 160 bits. Assume that \( F_q \) is a finite field of the module \( q \), where \( q \) is a large prime number. The elliptic curve over a finite field \( F_q \) can be defined as follows: \( E : y^2 = x^3 + ax + b \) (mod \( p \)), where \( a, b, x, y \in F_q \) and \( \Delta = 4a^3 + 27b^2 \neq 0 \) (mod \( p \)).

Definition 2 (Addition of elliptic curves). Assume that \( P = (x_1, y_1) \in E \), where \( P \) is a point of the elliptic curve \( E \) and \( -P = (x_1, -y_1) \) (mod \( p \)) is the negative point of \( P \). Suppose \( Q = (x_2, y_2) \in E \), \( Q \neq -P \); we can define a line \( l \) passes through \( P \) and \( Q \), and intersects the elliptic curve at a point \( R' = (x_3, y_3) \). The symmetrical point about the x-axis with \( R' \) is \( R = (x_3, -y_3) \); then we can define \( R = P + Q \). In addition, scalar multiplication operation on the elliptic curve can be
described as follows:
\[ k \cdot P = P + P + P + \cdots + P \quad (k \in \mathbb{Z}_q^*) \]  

**Definition 3** (Elliptic curve discrete logarithm problem). Assume that \( P_1 \) is a point on the elliptic curve \( E \) on the finite field \( F_p \), and select a random number \( k \in \mathbb{Z}_q^* \). Then, we can calculate \( P_2 = k \cdot P_1 \). In this case, there is the feasibility of the calculation of \( P_2 \) according to Definition 2. According to the elliptic curve discrete logarithm problem (ECDLP), however, it is hardly possible to get \( k \) according to the above equation.

3.2. Forking Lemma

**Definition 4** (Forking lemma [18]). Suppose that \( A \) is a probabilistic polynomial time turing machine, and its input includes public data. We use \( Q \) and \( R \) to symbolize the number of queries that \( A \) can ask to the random oracle and the number of queries that \( A \) can ask to the signer, respectively. Suppose that over a period of time \( T \), \( A \) can generate a legitimate signature \( (m, \sigma_1, h, \sigma_2) \) within probability \( \epsilon \geq 10(R + 1)(R + Q)/2^k \). If someone does not know the private key, but successfully forge the signature \( (\sigma_1, h, \sigma_2) \) with an indistinguishable distribution probability, then we can imagine a machine, which can get the secret information from the machine and obtain and replace the interaction with the signer by simulation. Eventually, it can generate two legitimate signatures \( (m, \sigma_1, h, \sigma_2) \) and \( (m, \sigma_1', h', \sigma_2') \) such that \( h \neq h' \) in expected time \( T' \geq 120686QT/e \).

3.3. Certificateless (Aggregate) Signature Scheme. Generally, a certificateless signature (CLS) scheme and a certificateless aggregate signature (CL-AS) scheme consist of the following seven algorithms.

1. **Setup**: the KGC and TA will execute this probabilistic algorithm, which needs a security parameter \( \lambda \), then generates an elliptic curve \( E \), public keys PKTA and \( PK_{KGC} \), and master secrets key \( a, \beta \), respectively, publishes a number of system parameters which is used for ensuring the system in order.

2. **PartialPrivateKeyGeneration**: in this algorithm, firstly, the entity \( V_i \) transmits a tuple which includes its real identity and partial pseudo identity to TA. Then TA sends a whole pseudo identity to KGC with calculation. Eventually, KGC transmits the partial private key to entity \( V_i \) in a secure channel.

3. **VehicleKeyGeneration**: the entity \( V_i \) selects random \( r_i \in \mathbb{Z}_q^* \) as its secret key and calculates its public key \( PK_{V_i} \).

4. **IndividualSign**: this algorithm is used by each entity \( V_i \); after generating a message \( m_i \), the entity \( V_i \) tries to calculate a set of variables. Then it sends the signature \( \sigma \) to the verifier.

5. **IndividualVerify**: this algorithm is executed by the verifier such as RSU. When receiving input including signature \( \sigma \), pseudo identity \( PID_i \), and current time \( T_{cur} \), the RSU will check the time validity firstly. Then the algorithm will output true if the signature is valid or false otherwise.

6. **AggregateSign**: in this algorithm, generally the aggregate signature generator is RSU in our system. For an aggregating set \( V \) of \( n \) entities \( V_1, V_2, \ldots, V_n \), the pseudo identity \( PID_i \) of each vehicle \( V_i \) as list \( PID_i \), the corresponding public key \( PK_{V_i} \), and message signature tuples \( (m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_n, \sigma_n) \) from \( V_i \), respectively. The aggregate signature generator will generate signature \( \sigma \); then it will transmit the tuple including the signature, the list \( PID_i \), and time list \( T \) to the verifier.

7. **AggregateVerify**: in general, this algorithm is executed by another RSU. It takes an aggregating set \( V \) of \( n \) entities \( \{V_1, V_2, \ldots, V_n\} \), the pseudo identity \( PID_i \) of each entity \( V_i \). The verifier will check the time validity for each entity firstly. Then it will output true if the signature is valid or false otherwise.

3.4. Security Model. In this section, we will demonstrate the security model of CLS and CL-AS schemes. We consider two different types of adversaries: Type 1 \( \mathcal{A}_1 \) and Type 2 \( \mathcal{A}_2 \). To be specific, adversary \( \mathcal{A}_1 \) is able to replace a user’s public key or private key but cannot access or even replace the master secret key of KGC. And adversary \( \mathcal{A}_2 \) is able to access the master secret key of KGC, which can be called an internal attacker. However, it cannot replace or access the public key of a certain user.

Generally, we use two games to model the security of CLS and CL-AS schemes, which is played between an adversary \( \mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\} \) and a challenger \( \mathcal{C} \). \( \mathcal{C} \) can access five oracles to get what he needs. The details are as follows:

1. **GenerateUser**: given a user’s ID \( PID_i \) and request for its public key \( PK_{V_i} \), \( \mathcal{C} \) returns the public key \( PK_{V_i} \) of \( PID_i \).

2. **RevealPartialPrivateKey**: given a user’s pseudo identity \( PID_i \), \( \mathcal{C} \) outputs the corresponding partial secret key \( PPK_{V_i} \).

3. **RevealSecretKey**: given a user’s pseudo identity \( PID_i \), \( \mathcal{C} \) submits the user’s secret key \( r_i \).

4. **ReplaceKey**: given a user’s pseudo identity \( PID_i \) and the public key \( PK_{V_i} \), \( \mathcal{C} \) will replace the public key \( PK_{V_i} \) with \( PK_{V_i} \).

5. **Sign**: given a message \( m_i \in \{0, 1\}^* \), \( \mathcal{C} \) uses the algorithm to generate a signature \( \sigma \) corresponding to user \( PID_i \) on message \( m_i \) and submits it to \( \mathcal{A} \).

We construct the following two games, Game I and Game II, for our schemes:
Step 1. $C$ runs the Setup algorithm to generate a master secret key $\beta$, a list of system parameters, and the system public key $PK_{TA}$. It then sends the system parameters to $A_1$ and keeps $\beta$ secret.

Step 2. $A_1$ queries the GenerateUser, RevealPartialSecretKey, RevealSecretKey, and Sign oracles in order.

Step 3. $A_1$ generates the corresponding public key $PK_{V_i}$ and a signature $\sigma_i$ of a user with identity $PID_i$.

$A_1$ will win the game if the following conditions are met:

(i) It neither uses PID$_i$ to access the RevealPartialSecretKey query nor obtains the partial private key

(ii) $\sigma_i$ is a valid signature of the user with the identity PID$_i$ and the corresponding public key PK$_{V_i}$

(iii) It never uses (PID$_i$, $m_i$) to query the Sign oracle

(Game II) A Type 2 adversary $A_2$ and a challenger $C$ will try to play the game as follows:

Step 1. $C$ runs the Setup algorithm to generate a master secret key $\beta$, a list of system parameters, and the system public key $PK_{TA}$. It then sends the system parameters to $A_2$. $A_2$ will win the game if the following conditions are satisfied:

(i) It never use PID$_i$ to access the RevealSecretKey or ReplaceKey query to obtain the partial private key

(ii) $\sigma_i$ is a valid signature of user with identity PID$_i$ and the corresponding public key PK$_{V_i}$

(iii) It never uses (PID$_i$, $m_i$) to query the Sign oracle

Definition 5. The CLS scheme is provably secure, if neither polynomial time adversary $A_1$ or $A_2$ is able to win Game I and Game II, respectively with a non-negligible advantage.

We construct the following two games, Game III and Game IV, for our CL-AS scheme.

(Game III) A Type 1 adversary $A_1$ and a challenger $C$ will try to play the game as follows:

Step 1. $C$ runs the Setup algorithm to generate the master secret key $\beta$, system parameter, and the system public key $PK_{TA}$. It then sends the system parameter to $A_1$ and keeps $\beta$ secret.

Step 2. $A_1$ queries the GenerateUser, RevealPartialSecretKey, RevealSecretKey, and Sign oracles in order.

Step 3. $A_1$ outputs an aggregate signature $\sigma^*$ of $n$ users with identity $PID^* = \{PID_{1}, PID_{2}, \ldots, PID_{n}\}$ and the corresponding public key $PK^*_V = \{PK_{V_{1}}, PK_{V_{2}}, \ldots, PK_{V_{n}}\}$ on messages $m^* = \{m_{1}, m_{2}, \ldots, m_{n}\}$.

$A_1$ wins the game if the following conditions are satisfied:

(i) At least one of the identities has not been submitted to the RevealPartialSecretKey query to obtain the partial secret key

(ii) $\sigma^*$ is a valid signature on $n$ messages $M^* = \{m_{1}, m_{2}, \ldots, m_{n}\}$ of $n$ users with identities $PID^* = \{PID_{1}, PID_{2}, \ldots, PID_{n}\}$ and the corresponding public key $PK^*_V = \{PK_{V_{1}}, PK_{V_{2}}, \ldots, PK_{V_{n}}\}$

(iii) It never uses (PID$_i$, $m_i$) to query the Sign oracle

(Game IV) A Type 2 adversary $A_2$ and a challenger $C$ will try to play the game as follows:

Step 1. $C$ runs the Setup algorithm to generate the master secret key $\beta$, system parameter, and the system public key $PK_{TA}$. $A_2$ will win the game if the following conditions are satisfied:

(i) It has not used all of the identities to access the RevealPartialSecretKey query to obtain the partial private key.

(ii) $\sigma^*$ is a legitimate signature on $n$ messages $m^* = \{m_{1}, m_{2}, \ldots, m_{n}\}$ of $n$ users with identities $PID^* = \{PID_{1}, PID_{2}, \ldots, PID_{n}\}$ and the corresponding public key $PK^*_V = \{PK_{V_{1}}, PK_{V_{2}}, \ldots, PK_{V_{n}}\}$

(iii) It never uses (ID$_i$, $m_i$) to query the Sign oracle
**Definition 6.** The CL-AS scheme is provably secure, if neither polynomial time adversary \( \mathcal{A}_1 \) or \( \mathcal{A}_2 \) is able to win Game III and Game IV, respectively, with a nonnegligible advantage.

### 4. Overview of Kamil et al.’s CLS and CL-AS Scheme

In the scheme proposed by Kamil et al. [6], there mainly exist four entities including TA, regional transport management authority (RTMA), which is a trusted party responsible for partial secret key generation, RSU, and vehicle. The scheme is reviewed as follows:

#### 4.1. Overview of Kamil et al.’s CLS Scheme

1. **Setup:** the TA selects a security parameter \( k \), two secure primes \( p \) and \( q \), an elliptic curve \( E \) which can be defined by the equation \( y^2 = x^3 + ax + b \mod p \), where \( a, b \in \mathbb{F}_p \), a generator \( P \) with order \( q \) of additive group \( G \) consisting of all the points on \( E \), and five hash functions, \( h_0, h_1, h_2, h_3, \) and \( h_4 \). Then, it picks \( x \in \mathbb{Z}_q^* \) as its master secret key and calculates its public key \( P_{pub} \). Also, TA defines a time-function \( f(t) \), where \( t \) is the current time. TA publishes the \( \text{param} = \{ p, q, a, b, P, P_{pub}, h_0, h_1, h_2, h_3, h_4, f(t) \} \).

2. **UserRegistration:** the RTMA executes the following algorithm to register a vehicle with an identity \( ID_k \). Firstly, vehicle sends its identity \( ID_k \) to the RTMA. Then RTMA randomly selects \( h_{y,k} \in \mathbb{Z}_q^* \) and calculates hash chain set \( h_{y,k} = \{ h_{y,k}, h_{2y,k}, \ldots, h_{ny,k} \} \), \( 1 \leq y \leq n \), where \( h_{y,k} = h_{y}(h_{y-1,k}) \).

3. **PartialSecretKeyGeneration:** after receiving param and a vehicle with identity \( ID_k \), RTMA runs as follows:

4. **PseudonymGeneration:** after receiving the tuple \( (h_{y,k}, (A_k, x_k)) \) from the RTMA, the vehicle executes the following algorithm:

5. **UserKeyGeneration:** vehicle with \( ID_k \) uses the algorithm to generate its private key:

6. **Sign:** after receiving \( \text{param} \), PSK_{A_k}, SK_{A_k}, and PK_{A_k}, a vehicle with pseudo identity \( PID_{y,k} \) can sign on a message \( m_k \) as follows:

7. **Verify:** after receiving the tuple \( (PID_{y,k}, m_k, PK_{A_k}, \omega_k, \sigma_k, T_k) \), verifier can use the algorithm to verify any signature with following steps:

**Step 1.** The RTMA generates its public key \( PK_{RTMA} = s \cdot P \), where secret key \( s \in \mathbb{Z}_q^* \) is randomly selected.

**Step 2.** Calculate \( \alpha_k = h_2(\text{param})P_{pub} || ID_k \), \( \beta_k = h_2(ID_k || \text{D}_{RTMA} || \text{V}, t = h_2(V) \), and \( \xi_k = h_2(ID_k || PK_{RTMA}) \).

**Step 3.** Compute \( A_k = t \beta_k \cdot P + x_k = t \beta_k + \xi_k \alpha_k s \).

**Step 4.** Publish \( PK_{RTMA} \), send \( (PSK_{A_k} = (A_k, x_k), h_{y,k}) \) to the vehicle and \( (h_{y,k}, ID_k) \) to TA.

**Step 1.** Compute \( \alpha_k = h_2(\text{param})P_{pub} || ID_k \), \( t = h_2(V) \), and \( \xi_k = h_2(ID_k || PK_{RTMA}) \).

**Step 2.** Check \( PSK_{A_k} \) is valid or not with the equation holds.

\[
x_k \cdot P = A_k + \xi_k \alpha_k \cdot PK_{RTMA}.
\]

**Step 3.** Compute its pseudonym set as \( \{ PID_{1,k}, PID_{2,k}, \ldots, PID_{y,k} \} \) at timeslot \( ts_{cur} \), where \( PID_{y,k} = h_1(ID_k || h_{y,k} || V || ts_{cur}) \).

**Step 1.** Choose \( a_k, r_k, r_k^2 \in \mathbb{Z}_q^* \) in random.

**Step 2.** Calculate \( SK_1^k = h_3(r_k^2 || A_k || PID_{y,k}) \) and \( SK_2^k = h_3(r_k^2 || x_k || PID_{y,k}) \).

**Step 3.** Output \( SK_k = a_k(SK_1^k + SK_2^k) \) and \( PK_k = SK_k \cdot P \) as its private and public keys, respectively.

**Step 1.** Randomly pick \( d_k \in \mathbb{Z}_q^* \) and calculate \( \omega_k = \xi_k \alpha_k \).

**Step 2.** Calculate \( v_k = h_4(PID_{y,k} || m_k || SK_1^k || SK_2^k || T_k) \), \( h_k = h_4(PID_{y,k} || m_k || \omega_k || PK_k || PK_{RTMA} || P_{pub} || T_k) \), and \( \delta_k = h_4(m_k || PK_k || V || T_k) \).

**Step 3.** Calculate \( y_k = d_k v_k, \Omega_k = y_k \cdot P, R_k = \delta_k \cdot PK_k + h_k \cdot A_k + \Omega_k, \text{ and } \forall \cdot \delta_k SK_k + h_k x_k + d_k v_k \).

**Step 4.** Output signature \( \sigma_k = (R_k, \forall \cdot k) \) on message \( m_k \) and transmits \( (PID_{y,k}, m_k, PK_k, \omega_k, \sigma_k, T_k) \), where \( T_k \) is the current timestamp.

**Step 1.** Check whether the time delay equation holds. If it holds, then \( T_k \) is valid and it will accept the signature; otherwise, it will reject it.

**Step 2.** Calculate \( h_k = h_4(PID_{y,k} || m_k || \omega_k || PK_k || PK_{RTMA} || P_{pub} || T_k) \).

**Step 3.** Check whether the following equation holds.

\[
\forall \cdot P = R_k + h_k \omega_k \cdot PK_{RTMA}.
\]

If this equation holds, then the signature is valid; otherwise, it will be discarded.
4.2. Overview of Kamil et al.’s CL-AS Scheme. The Setup, UserRegistration, PartialPrivateKeyGeneration, PseudonymGeneration, VehicleKeyGeneration, Sign, and Verify algorithms are the same as the above CLS scheme. In addition, the Aggregate and AggregateVerify algorithms are described as follows:

(1) Aggregate: in general, the roadside unit (RSU) acts as the aggregator. When receiving $n$ certificateless signatures \( \{ \sigma_i = (R_i, \mathcal{y}_i), \sigma_i = (R_2, \mathcal{y}_2), \ldots, \sigma_i = (R_n, \mathcal{y}_n) \} \) on messages \( \{ (m_1||T_1), (m_2||T_2), \ldots, (m_n||T_n) \} \) from $n$ pseudonyms \( \{ \text{PID}_{\alpha, 1}, \text{PID}_{\alpha, 2}, \ldots, \text{PID}_{\alpha, n} \} \) under the state information \( \mathcal{V} \), the RSU calculates \( \mathcal{V}' = \sum \mathcal{y}_k^k \text{ and } R = \sum R_k \), then outputs an aggregate certificateless signature \( \sigma_T = (R, \mathcal{V}') \).

(2) AggregateVerify: generally another RSU or AS acts as the verifier. When receiving a certificateless aggregate signature \( \sigma_T = (R, \mathcal{V}') \) signed by $n$ vehicles. Then it will run as follows:

Step 1. Check whether the timestamp $T_k$ is valid, if not, it aborts, and if it holds, it runs the following steps.

Step 2. Compute $h_k = h_i(\text{PID}_{\alpha, k}||m_1||\omega_k||\text{PK}_{\alpha}||\text{PK}_{\text{RTMA}}||P_{\text{pub}}||T_k)$.

Step 3. Check whether the following equation holds.

\[
\mathcal{V}' \cdot P = R + \sum_{k=1}^{n} h_k \omega_k \cdot \text{PK}_{\text{RTMA}},
\]

if it holds, it receives all the signatures; otherwise, the signature is rejected.

4.3. Cryptanalysis of Kamil et al.’s CL-AS Scheme. The security problem in the scheme proposed by Kamil et al. [6] mainly lies in the coalition attack, which is a kind of attack by a number of collusive vehicles.

As is described in Figure 1, in the coalition attack, two or more vehicles secretly change a part of their messages such as locations to hide their real locations and routes since the RSU (verifier) receives the exchanged signature. Then something of the collusive vehicles is exchanged officially. Which will definitely harm the system and even worse cause a serious accident.

We describe the coalition attack on Kamil et al.’s CL-AS scheme to illustrate its security flaws.

Assume that two users \( \{ U_1, U_2 \} \) have pseudonym \( \{ \text{PID}_{\alpha, 1}, \text{PID}_{\alpha, 2} \} \) and message \( \{ m_1, m_2 \} \), respectively. We show that two users can cooperate to generate valid aggregate signatures even if their individual signature is invalid. Two users can implement the coalition attack by executing the following algorithms.

Step 1. The user $U_i$ \( (i \in \{ 1, 2 \} ) \) randomly picks $d_i \in \mathbb{Z}_q^*$ and calculates $\omega_i = \xi \cdot \alpha_i$.

Step 2. Calculate $\beta_i = h_2(\text{ID}||\text{ID}_{\text{RTMA}}||\mathcal{V}), \delta_i = h_2(\text{ID}||\text{PK}_{\text{RTMA}})$, $x_i = t \beta_i + \xi \cdot \alpha_i, v_i = h_i(\text{PID}_{\alpha, i}||m_i||\text{SK}_{\alpha}||\text{SK}_{\delta}||T_i)$, $\delta_i = h_i(m_i||\text{PK}_{\alpha}||\mathcal{V}||T_i), y_i = d_i v_i, \Omega = y_i \cdot P$, and $R_i = \delta_i \cdot \text{PK}_i + h_i \cdot A_i + \Omega$.

Step 3. Then, $U_1$ sends $h_1 x_1$ to $U_2$; likewise, $U_2$ sends $h_2 x_2$ to $U_1$ in a secure channel. Then $U_1$ calculates $\mathcal{V}'_1 = \delta_1 \text{SK}_1 + h_2 x_2 + d_1 v_1$; likewise, $U_2$ calculates $\mathcal{V}'_2 = \delta_2 \text{SK}_2 + h_1 x_1 + d_2 v_2$.

Step 4. Eventually, they can output signature $\sigma_i = (R_i, \mathcal{V}'_i)$ and transmits \( \{ \text{PID}_{\alpha, i}, m_i, \text{PK}_i, \omega_i, \sigma_i, T_i \} \).

\[
\mathcal{V}' \cdot P = (\mathcal{V}'_1 + \mathcal{V}'_2) \cdot P = (\delta_1 \text{SK}_1 + h_2 x_2 + d_1 v_1 + \delta_2 \text{SK}_2 + h_1 x_1 + d_2 v_2) \\
\cdot P = \left( \sum_{k=1}^{2} \delta_k \text{SK}_k + h_i (t \beta_i + \xi \cdot \alpha_i) + d_i v_i \right) \\
\cdot P = \sum_{k=1}^{2} \delta_k \cdot \text{PK}_k + h_i \cdot A_i + \omega_k \\
+ \sum_{k=1}^{2} h_i \omega_k \cdot \text{PK}_{\text{RTMA}} = R + \sum_{k=1}^{2} h_i \omega_k \cdot \text{PK}_{\text{RTMA}}
\]

(5)

Obviously, the signature $\sigma_i = (R_i, \mathcal{V}'_i)$ is not a valid signature. However, when the RSU or AS aggregates the signature as $\sigma = (R_1 + R_2, \mathcal{V}'_1 + \mathcal{V}'_2)$, it will be a valid signature which satisfies the following equation.

Therefore, the above analysis shows that two malicious users can collude with each other to forge an aggregate signature. Actually, the coalition attack is originally caused by commutative law of addition. Similarly, $n$ users can also forge an aggregate signature with the same algorithms. Hence, Kamil et al.’s CL-AS scheme cannot resist coalition attacks.

5. Our Proposed CLS and CL-AS Schemes

5.1. System Model. In this section, we will try to describe our system model in detail including specific explanations. In order to be more specific, the system model is shown in Figure 2. There are four participants in total: trusted authority (TA), key generation center (KGC), road-side unit (RSU), and vehicle, which can be divided into two layers: the upper layer includes TA and KGC, and the lower layer consists of RSUs and vehicles. The demonstration of each participant is as follows:

(1) TA: it is a fully trusted third party that is responsible for system initialization, user registration, system parameter generation, and system security implementation. If necessary, it can track malicious behavior and catch malicious nodes. In addition, it also has enough computing power and storage capacity.
(2) KGC: it is a partially trusted party used for generating partial private key. It can help a vehicle generate partial secret key which contribute to its privacy security. Like the TA, it also has sufficient memory, processing, and computing capabilities.

(3) RSU: it is a smart application device installed in the roadside, which is able to transmit and submit information to TA, KGC, vehicles, or other RSUs in a secure wired connection. In addition, RSU commonly has limited computing power and storage capacity.

(4) Vehicle: it is the major and basic member in VANETs, which is generally equipped with a smart device which can perform the basic function such as transmitting the vehicle’s message and performing simple calculation. In addition, vehicle commonly has limited computing power and storage capacity.

Note that TA and KGC are functionally two completely different entities that can be deployed on a single server during deployment.

5.2. Design Requirements. For the safety of communication in VANETs, security and privacy are crucial. According to the latest research in this field, the proposed scheme for VANETs must satisfy the following security requirements:

(1) Message Integrity and Authentication: an eligible vehicle should be able to check that whether a message is sent and signed by a legitimate vehicle and is not forged or modified by the malicious entity.

(2) Identity Privacy Preservation: a vehicle should remain anonymous in all circumstances, which means that other malicious entities cannot infer its identity by taking and analyzing multiple pieces of messages about it.

(3) Traceability: the TA must have the ability to trace and obtain the vehicle’s real identity, even if the vehicle’s identity is anonymous.

(4) Unlinkability: a potentially malicious vehicle must not cross-link two messages sent by the same vehicle to prevent them from extrapolating the route of the vehicle from the information.

(5) Resistance to Attacks: a reasonable scheme should have the ability to withstand various general attacks such as the coalition attack, the impersonation attack, the modification attack, and the replay attack.
5.3. Our Proposed CLS Scheme. Our proposed CLS scheme includes five algorithms: Setup, PartialPrivateKeyGeneration, VehicleKeyGeneration, Sign, and Verify. The Notation to be used is listed in Table 1, and descriptions for algorithms are vividly shown in Figure 3 and described as follows:

(1) Setup: when given an appropriate security parameter \( \lambda \), TA will use the \( \lambda \) to generate and output the param by executing the following algorithms:

(2) PartialPrivateKeyGeneration: the algorithm will eventually generate the vehicle’s partial private key through the algorithms as follows:

(3) VehicleKeyGeneration: after receiving the partial private key \( PPK_i \), the vehicle \( V_i \) check if the equation \( PPK_i \cdot P = Q_i + n_i \cdot PK_{KGC} \) holds. If it holds, the partial private key \( PPK_i \) is valid. The vehicle randomly selects its private key \( \rho_i \in \mathbb{Z}^*_{q_i} \), then calculates its public key \( PK_{V_i} = \rho_i \cdot P \).

(4) Sign: in order to achieve authentication and message integrity, when the message is received by any entity, it has to be signed and verified. A vehicle \( V_i \) uses its pseudo identity \( PID_i \) and picks the latest timestamp \( TS \). The updated timestamp \( TS \) protects a signed message against replay attacks. Given the signing key \( (PPK_i, PK_{V_i}) \) and a traffic related message \( m_i \), the vehicle \( V_i \) performs the following steps, which are repeated every 100 – 300 ms in accordance with DSRC protocol [20]:

Table 1: List of notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>Trusted authority</td>
</tr>
<tr>
<td>KGC</td>
<td>Key generation center</td>
</tr>
<tr>
<td>RSU</td>
<td>Road-side unit</td>
</tr>
<tr>
<td>( h_i(\cdot), i = 1,2,3 )</td>
<td>One-way collision-resistant hash function</td>
</tr>
<tr>
<td>( p,q )</td>
<td>Two secure prime numbers</td>
</tr>
<tr>
<td>( E )</td>
<td>An elliptic curve ( y^2 = x^3 + ax + b \mod p )</td>
</tr>
<tr>
<td>( G )</td>
<td>An additive group, the order of which is ( q )</td>
</tr>
<tr>
<td>( P )</td>
<td>A generator of the group ( G )</td>
</tr>
<tr>
<td>( (PK_{TA}, \alpha) )</td>
<td>The public key and private key of the TA</td>
</tr>
<tr>
<td>( (PK_{KGC}, \beta) )</td>
<td>The public key and private key of the KGC</td>
</tr>
<tr>
<td>( (PK_{V_i}, \rho_i) )</td>
<td>The public key and private key of the vehicle ( V_i )</td>
</tr>
<tr>
<td>( PPK_i )</td>
<td>Partial private key of the vehicle ( V_i )</td>
</tr>
<tr>
<td>( PID_i )</td>
<td>Pseudo identity of the vehicle ( V_i )</td>
</tr>
<tr>
<td>( TS )</td>
<td>The latest timestamp</td>
</tr>
<tr>
<td>( V )</td>
<td>State information</td>
</tr>
</tbody>
</table>
(5) Verify: when an RSU or other entity receives the signature $\sigma$ and the tuple $(Q_i, PID_i, m_i, PK_{V_i}, TS)$ from the vehicle $V_i$, it can execute the algorithms to verify the message as follows:

**Step 1.** Firstly, select two secure prime numbers $p$ and $q$, then choose $a, b \in F_p$, which generate an elliptic curve $E$ defined by the equation $y^2 = x^3 + ax + b \mod p$, where $\Delta = 4a^3 + 27b^2 \neq 0 \mod p$ and generator $P$ of the additive group $G$ consisting of all the points on $E$.

**Step 2.** Choose $\alpha \in \mathbb{Z}_q^*$ in random, which serves as the master secret key and computes master public key $PK_{TA} = \alpha \cdot P$.

KGC selects $\beta \in \mathbb{Z}_q^*$ in random, then calculates $PK_{KGC} = \beta \cdot P$, which is the public key of KGC.

**Step 3.** Select three secure hash functions in random: $h_1: G \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $h_2: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $h_3: \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$.

**Step 4.** Store its master secret key $\alpha$ in its repository and keep it safe. Then publish all the system parameter:

$$\text{param} = \{p, q, a, b, P, PK_{TA}, PK_{KGC}, h_1, h_2, h_3\}. \quad (6)$$

**Step 1.** The vehicle $V_i$ with its real identity $\text{RID}_i$ randomly selects $x_i \in \mathbb{Z}_q^*$ as its private key and calculates its partial pseudo identity $\text{PID}_i = x_i \cdot P$. Then vehicle $V_i$ transmits $(\text{RID}_i, \text{PID}_i)$ to TA.

**Step 2.** After receiving the tuple, TA calculates another pseudo identity $\text{PID}_i^2 = \text{RID}_i \cdot h_1(\alpha \cdot \text{PID}_i^1 | T_{\text{cur}} | PK_{TA} | V)$, where $V$ is the system state information [19]; then TA sends the vehicle’s pseudo identity $\text{PID}_i = (\text{PID}_i^1, \text{PID}_i^2, T_{\text{cur}})$ to KGC in a secure way.

**Step 3.** KGC calculates $Q_i = y_i \cdot P$, $n_i = h_2(\text{PID}_i^3 | Q_i | V)$ and the vehicle’s partial private key $PPK_i = y_i + h_2(\text{PID}_i^3 | Q_i | V) \times \beta \mod p$. At last, KGC transmits the tuple $(Q_i, PPK_i)$ to vehicle $V_i$.

**Step 1.** Choose a random number $l_i \in \mathbb{Z}_q^*$ and calculate $L_i = l_i \cdot P$.

**Step 2.** Calculate $r_i = h_3(m_i | \text{PID}_i | V | PK_{V_i} | L_i | TS)$, where timestamp $TS$ is used to confirm time, and $S_i = r_i l_i + \rho_i + PP K_t \mod p$.

**Step 3.** The signature on message $m_i$ is $\sigma = (L_i, S_i)$; then the vehicle transmits the signature $\sigma$ and $(Q_i, PID_i, m_i, PK_{V_i}, TS)$ to the verifier.

**Step 1.** Check whether the TS is valid, if not, the algorithm aborts; otherwise, execute the next step.

**Step 2.** Calculate $r_i = h_3(m_i | \text{PID}_i | V | PK_{V_i} | L_i | TS)$ and $n_i = h_2(\text{PID}_i^3 | Q_i | V)$

**Step 3.** Check whether the following equation holds or not; if it holds, then the RSU or other entity will
accept the signature and the massage; otherwise, it will reject the message.

5.4. Our Proposed CL-AS Scheme. The Setup, Partial-Private-KeyGeneration, VehicleKeyGeneration, Sign, and Verify algorithms of CL-AS are similar to the proposed CLS scheme. In addition, the Aggregate and AggregateVerify are described as follows. Note that the Aggregate andAggregateVerify algorithms are usually executed by the same RSU to transmit less data in the communication process.

(1) Aggregate: when an aggregator such as a RSU receives \( n \) vehicles’ messages \( M = \{m_1, m_2, \ldots, m_n\} \), signatures \( \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \), timestamps \( TS = \{TS_1, TS_2, \ldots, TS_n\} \), \( Q = \{Q_1, Q_2, \ldots, Q_n\} \), public key of each vehicle \( PK_V = \{PK_{V_1}, PK_{V_2}, \ldots, PK_{V_n}\} \), \( L = \{L_1, L_2, \ldots, L_n\} \), and pseudo identities \( PID = \{PID_1, PID_2, \ldots, PID_n\} \). It can execute the following algorithms to aggregate the signature:

\[
S = \sum_{i=1}^{n} a_i S_i = \sum_{i=1}^{n} a_i (Q_i + PK_{V_i})
\]

5.5. Correctness of Individual Message Verification. The individual verification in the proposed scheme is correct. The correctness proof is as follows:

\[
S \cdot P = (r_i l_i + \rho_i + PPK) \cdot P = (r_i l_i + \rho_i + y_i + n_i \beta) \\
\cdot P = r_i l_i \cdot P + \rho_i \cdot P + y_i \cdot P + n_i \beta
\]  

(9)

5.6. Correctness of Aggregate Message Verification. The aggregate verification in the proposed scheme is correct. The correctness proof is as follows:

\[
S \cdot P = \sum_{i=1}^{n} a_i S_i \cdot P = \left( \sum_{i=1}^{n} a_i r_i l_i + a_i \rho_i + a_i PPK \right) \\
\cdot P = \sum_{i=1}^{n} a_i (r_i l_i \cdot P + \rho_i \cdot P + y_i \cdot P + n_i \beta \cdot P)
\]

(10)

5.7. Security Proof of the Proposed CLS Scheme. According to Definition 3, it is extremely hard to solve ECDLP. Therefore, we can prove that our CLS scheme is able to enforce nonforgerystand alone. On the basis of Definition 4, assume that a probabilistic polynomial-time forger \( A \) can forge a signature with an advantage \( \epsilon \). In addition, \( q_h \) denotes random oracles \( h_1 \), for \( i = 2, 3 \), \( q_{GU} \) denotes the Generate-User oracle, \( q_{PPK} \) denotes Partial-Private-Key oracle, and \( q_{SK} \) denotes the Secret-Key oracle. Then, we can know that a challenger \( C_1 \) can solve ECDLP during a time scope \( T \), when \( T \leq 10^{16} \) if \( \epsilon \geq 10(q_{GU} + 1)(q_{PPK} + q_{GU} + q_{SK} + q_{S})/q \).

(1) Setup: \( C_1 \) chooses \( \alpha \) and calculates \( PK_{TA} = \alpha \cdot P \) which serves as its private key and master public key. Then, \( C_1 \) will generate the system parameters \( \text{param} = (P, q, E, G, h_1, h_2, h_3, PK_{TA}, PK_{KGC}) \), and transmit it to \( A_1 \).

(i) \( h_3 \) Hash Query: \( C_1 \) will examine whether the hash list \( h_3 \) has the corresponding tuple if it receives the query with parameter \( (PID, Q) \) from \( A_1 \). If not, \( C_1 \) will select a random number \( r_{h_3} \), put it in the hash list \( h_3 \). If so, it needs to transmit \( r_{h_3} = h_2(PID, Q, V) \) to \( A_1 \).

(ii) \( h_3 \) Hash Query: \( C_1 \) will examine whether the hash list \( h_3 \) has the corresponding tuple \( (m_i, PID, PK_V, Z, TS, r_{h_3}) \) if it receives the query with parameter \( \text{param} = (m_i, PID, PK_V, L_i, TS) \) from \( A_1 \). If not, \( C_1 \) will choose a random number \( r_{h_3} \), put the tuple \( (m_i, PID, PK_V, L_i, TS, r_{h_3}) \) in the list \( h_3 \). If so, it will transmit \( r_{h_3} = h_2(m_i, PID, V) \) to \( A_1 \). Eventually, \( C_1 \) will transmit \( r_{h_3} = h_3(m_i, PID, V, V) \) to \( A_1 \).
Partial-Private-Key Query: after receiving a query about the identity PID, C1 will calculate Qt = y' · P, where y' is randomly selected, and check whether the hash list Lh has the corresponding tuple (PID, Qt, τh). If so, C1 will calculate PKi = y' + h2(PID, [Q], [V]) (mod p) and transmit the partial private key of vehicle V, PK, to A1. If not, it will halt.

User-Generation Query: suppose that the query is on the basis of the pseudo identity PID:

(i) C1 will check whether PKV exists in the list L, if the list L includes (PID, PKV, ρi). If not, a random number ρi ∈ Zq will be selected and C1 will calculate PKV = ρi · P. If so, it will transmit PKV, to A1. Eventually, the challenger C1 will transmit PKV to A1 and update the list.

(ii) C1 will set PKV = LL if the tuple (PID, PKV, ρi) does not exist in the list L. Then, a random number ρi ∈ Zq will be chosen and PKV = ρi · P will be calculated and ρi will be regarded as a private key. Eventually, C1 will transmit PKV, to A1 and put the tuple (PID, PKV, ρi) in the list L.

Private-Key Query:

(i) C1 will check whether ρi exists in the list L, if the list L includes (PID, PKV, ρi). If not, it will access a User-Generation query to output the public key PKV = ρi · P. Eventually, the challenger C1 will transmit ρi to A1 and update the list.

(ii) C1 will access a User-Generation query if the tuple (PID, PKV, ρi) does not exist in the list L. Eventually, C1 will transmit ρi to A1 and put the tuple (PID, PK, KV, ρi) to the list L.

Sign Query: after receiving a legitimate query about the message m1 of pseudo identity PID, C1 will check the tuple (PID, Qh, τh) in the hash list LH. Hence, C1 can easily check the value τh from the tuple and select two random numbers i1 and i2. Then, C1 will choose another two random numbers s1 and s2. Furthermore, C1 will calculate Zi = s1 · P − n1 · PK, and s2 = s1. Eventually, it will transmit (Lh, Sj) to A1 and put the tuple (m1, PID, PKV, L, TS, τh) in the list LH.

Theorem 7. According to the random oracle, when faced with an adaptive chosen message attack, our proposed scheme has the capacity of unforgeability.

Proof. Assume that an ECDLP sample (P, Q = x · P) is given, the elliptic curve E holds two points P and Q, and an adversary A1 is able to forge message (PID, PKV, m1, TS, σ3). Hence, we start a game between a challenger C1 and the adversary A1, which can execute and manipulate A1 to solve ECDLP with a nonnegligible probability.

We know the forking lemma in Definition 4 and apply it to our proposed scheme. After using the same random elements to replay A1, C1 succeeds in getting two legitimate signatures σ3 = (Z2, Sj) and σ'3 = (L2, S3) during a polynomial time period, where Sj = ri · Li + Qi + PKV (mod p) and S3 = ti · Li + Qi + PKV (mod p) by computing.

\[
t_i S_i - r_i S'_i \mod p = \frac{t_i r_i l_i + t_i Q_i + t_i PKV - r_i l_i - r_i PK - r_i Q_i + r_i PK}{t_i - r_i} \mod p = Q_i + PKV.
\] (11)

In conclusion, if ε ≥ 10(q1, + 1)(q2, + 1)(q3, + q4, + q5, + q6, + q7,)/q, then C1 is able to break the ECDLP during a time period which is less than 120686671ε. However, this conclusion is inconsistent with the difficulty of solving the ECDLP. Therefore, we can define that our proposed CLS scheme can resist a forgery attack.

5.8 Security Proof of the Proposed CL-AS Scheme. According to Definition 3, it is extremely hard to solve ECDLP. Therefore, we can prove that our scheme is able to enforce nonforgery. Furthermore, we will prove that our CL-AS scheme can also resist coalition attack.

(1) Setup: a random number α is selected as the master secret key, and the public key can also be calculated as PK = α · P. Then, the oracle simulation is ready to run. In this whole game, C2 maintains a list L = {PID, PKV, K, ρi} and responds to A2's oracle as follows.

(i) h2 Query: after receiving a pseudo identity PID, C2 will throw a coin c, where 0 holds a probability ε, and 1 holds a probability 1 − ε, then C2 will select...
According to the random oracle, when faced with an adaptive chosen message attack, our proposed CL-AS scheme has the capacity of unforgeability.

Then, $A_2$ will transmit $n$ vehicles with identities from the list $L_{PID} = \{PID_1, PID_2, \ldots, PID_n\}$, public keys from the list $L_{PKV} = \{PK_{V_1}, PK_{V_2}, \ldots, PK_{V_k}\}$, $n$ messages from the list $M = \{m_1, m_2, \ldots, m_n\}$, and a random list $R_\sigma = \{a_1, a_2, \ldots, a_n\}$, and a certificateless aggregate signature $\sigma^* = \{L', S'\}$. At the beginning, $C_2$ will select the $n$ tuples $(PID_i', u_i^w, c_i', Q_i')$ for $i = 1, 2, \ldots, n$ in the list $L_{h_3}$ and precede only $c_i = 1$ and $c_i = 0$ for $j = 1, 2, \ldots, n$, $j \neq k$. Note that the Sign oracle has not received the tuple $(PID'_2, PK_{V_2}' , m_2')$. Otherwise, $C_2$ will halt and fail. This success case signifies that $Q_h = u_{k}^{w} \cdot PK_{TA}$ and $Q_j = u_{j}^{w} \cdot P$ for $j = 1, 2, \ldots, n, j \neq k$. In addition, the aggregate signature $\sigma^* = \{L', S'\}$ is supposed to satisfy the aggregate verification equation $S \cdot P = \sum_{i=1}^{n} (a_i r_i \cdot L_i) + Z + \sum_{i=1}^{n} a_i n_i \cdot PK_{KGC}$.

Accordingly, $C_2$ checks the tuples $(m_i', PID_i', PK_{V_i}', Z_i')$, $u_i^w$ in the list $L_{h_3}$ and the tuple $(PID_i', PPK_i, PK_{V_i}, \rho_i)$ from $L$. Later, it calculates $S_i' = u_{i}^{w} \cdot a_i \mod p$, which will satisfy $S_i' \cdot P = u_{i}^{w} \cdot PK_{TA} = Q_i'$ for $i = 1, 2, \ldots, n, i \neq k$. Eventually, $C_2$ constructs $S_i'$ as $S_i' = S_i' = S_i - \sum_{i=1}^{n} a_i n_i$, and $S_i' = PK_{KGC}' + \sum_{i=1}^{n} u_i^w r_i$ (mod $p$) for $L'_{i} = l_{i}'$ and $l_{i}' \in Z_q$ for $1 \leq i \leq n$. $C_2$ will select a random number $h_{k}^{*} \in Z_q$ and calculate $Z_i = (h_{k}^{*})^{-1} \sum_{i=1}^{n} u_i^w a_i (r_i \cdot L_i + Q_i + PK_{V_i})$.

Hence, the hash value $h_k(m_i', PID_i', PK_{V_i}', Z_i')$ is defined as $h_k^*$. It will use $h_k^*$ until it does not repeat if the list $L_{h_3}$ holds the tuple $(m_i^*, PID_i^*, PK_{V_i}', Z_i^*)$. Consequently, the signature $(L'^*, S'^*)$ is a legitimate certificateless signature on message $m_i^*$ for the reason that the equation below:

\[
Q_i^w + h_{k}^{*} + h_{k}^{*} Z_i'^* = Q_i^w
\]

\[
= Q_i^w + h_{k}^{*} + \sum_{i=1}^{n} u_i^w a_i (r_i \cdot L_i + Q_i + PK_{V_i})
\]

\[
= PPK_i \cdot P + \sum_{i=1}^{n} u_i^w a_i (r_i \cdot L_i + Q_i + PK_{V_i}) = S'^* \cdot P.
\]

Eventually, $S$ can get the signature $(L'^*, S'^*)$ as a forgery of the certificateless signature scheme. However, this conclusion is inconsistent with the difficulty of solving the ECDLP. Therefore, we can define that our proposed CLS scheme can resist a forgery attack.

**Theorem 8.** The proposed certificateless aggregate signature (CL-AS) scheme can resist coalition attacks.

**Theorem 9.** The proposed certificateless aggregate signature (CL-AS) scheme can resist coalition attacks.
6. Performance and Security Analysis

6.1. Security Analysis

(1) Traceability: in the proposed scheme, only TA has the real identity of the certain vehicle. After submitting the pseudo identity \( \text{PID}_i = (\text{PID}_i^1, \text{PID}_i^2, T_{\text{cur}}) \), TA can easily trace back to the vehicle’s real identity \( \text{RID}_i \) in accordance with the equation \( \text{RID}_i = \text{RID}_i^1 \oplus h_1(\text{aPID}_i^1 \| T_{\text{cur}} \| \text{PK}_{\text{TA}} \| \text{V}) \). Therefore, according to the RID list, TA can trace back to the certain vehicle \( V_i \), even revoke it. (\( \text{RID}_i, \text{PID}_i^1 \))

(2) Message integrity and authentication: according to Definition 3, the ECDLP problem is hard, so that no polynomial adversary can forge a valid message. Therefore, the verifier can verify the validity and integrity of the message \( \{ Q_i, \text{PID}, \text{PK}_V, m_i, T, \sigma_i \} \) by verifying whether the equation \( S_i \cdot P = r_f \cdot L_i + P \cdot \text{PK}_V + Q_i + n_i \cdot \text{PK}_{\text{KGC}} \) holds or not. Therefore, our proposed scheme for VANETs provides message authentication and integrity.

(3) Resistance to replay attacks: the proposed scheme can resist the replay attack for the reason that the tuple \( \{ Q_i, \text{PID}, \text{PK}_V, m_i, T, \sigma_i \} \) includes the timestamp \( T \). RSU and other vehicles will check the validity of the signature, so they are able to detect the replay of the message. Hence, our proposed scheme for VANETs can resist replay attacks.

(4) Resistance to coalition attacks: our proposed scheme can resist the coalition attacks, because we improve the signature generation process. To be specific, we choose a random list to change the ratio in the equation \( S = \sum_{i=1}^{n} a_i S_i \). Therefore, our scheme uses this method to resist the coalition attacks.

(5) Resistance to stealing of the check table: in the proposed scheme, TA, KGC, vehicles, and RSUs do not require a check list. Therefore, an attacker cannot complete an attack by stealing any checklist. Hence, the proposed scheme can resist the attack of the checklist.

6.2. Performance Analysis. In this section, we will discuss the performance of the proposed scheme and related schemes and make a comparison in detail. We adopt the method of computation evaluation where the bilinear pairing on the security level of 80 bits is created as follows: \( e : G_1 \times G_2 \rightarrow G_2 \), where \( G_1 \) is an additive group generated by a point \( P \) with order \( q \) on a super singular elliptic curve \( E : y^2 = x^3 + x \mod p \) with embedding degree 2, \( p \) is a 512-bit prime number, \( q \) is a 160-bit prime number [25]. The ECC on the security of 80 bits is constructed as follows: \( G \) is an additive group with order \( q \) that is generated on a nonsingular elliptic curve \( E : y^2 = x^3 + ax + b \mod p \), where \( p, q \) are 160-bit primes and \( a, b \in \mathbb{Z}_p^* \). The experiment is conducted using the well-known python cryptographic library PyCryptodome on a desktop running Intel I5-9400 @ 2.90 GHz processor, with 16 GB memory running Windows 10 operating system. The notations of the cryptographic operations used in this paper and their running times are given in Table 2. Table 3 shows the summary of the computation costs in terms of signing a message, verifying a single message, and verifying \( n \) messages.

In [13, 24], their schemes choose to use bilinear pairing, which significantly increases their operation time. As a contrast, other four schemes [6, 12, 17, 23] do not use bilinear pairing, which can substantially reduce computation time.

In our scheme, \( L_i = l_i \cdot P \) uses a scalar multiplication in ECC operation and the calculation of \( r_i \) uses a one-way hash function during the individual sign process. In individual verification, we use three scalar multiplication operations for \( S_i \cdot P, r_i \cdot L_i \), and \( n_i \cdot \text{PK}_{\text{KGC}}, \) three addition operations, and two one-way hash function operations for the calculations of \( r_i \) and \( n_i \). In aggregate verification process, we use \( n + 2 \) scalar multiplication operations for \( \sum_{i=1}^{n} a_i r_i \cdot L_i, S \cdot P, \) and \( \sum_{i=1}^{n} a_i n_i \cdot \text{PK}_{\text{KGC}}, \) two addition operations, and \( 2n \) one-way hash function operations for each \( n_i \) and \( r_i \). By comparison, our scheme has low time complexity and high efficiency. In addition, our scheme can resist coalition attacks, which are a special and security feature that no other scheme has.

We use the data in Table 3 to generate three figures, which can intuitively compare other related schemes with

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{bp} )</td>
<td>Bilinear pairing operation</td>
<td>4.1603</td>
</tr>
<tr>
<td>( T_{bp,m} )</td>
<td>Scalar multiplication in bilinear pairing operation</td>
<td>1.6722</td>
</tr>
<tr>
<td>( T_{bp,a} )</td>
<td>Addition in bilinear pairing operation</td>
<td>0.0069</td>
</tr>
<tr>
<td>( T_{e,m} )</td>
<td>Scalar multiplication in ECC operation</td>
<td>1.1280</td>
</tr>
<tr>
<td>( T_{e,a} )</td>
<td>Addition in ECC operation</td>
<td>0.0339</td>
</tr>
<tr>
<td>( T_h )</td>
<td>One-way hash function</td>
<td>0.0360</td>
</tr>
<tr>
<td>Scheme</td>
<td>Individual sign</td>
<td>Individual verify</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Kamil et al. [6]</td>
<td>$3T_{e-m}+2T_{e-a}$ + $3T_h = 3.5598ms$</td>
<td>$2T_{e-m}+T_{e-a}+T_h = 2.6988ms$</td>
</tr>
<tr>
<td>Kumar et al. [12]</td>
<td>$4T_{e-m}+2T_{e-a}$ + $3T_h = 4.6878ms$</td>
<td>$4T_{e-m}+3T_{e-a}+4T_h = 4.7577ms$</td>
</tr>
<tr>
<td>Yang et al. [13]</td>
<td>$4T_{bp-m}+2T_{bp-a}$ + $3T_h = 6.9582ms$</td>
<td>$3T_{bp}+3T_{bp-m}+T_{bp-a}$ + $4T_h = 17.911ms$</td>
</tr>
<tr>
<td>Cui et al. [17]</td>
<td>$T_{e-m} + T_{e-a} + T_h = 1.1979ms$</td>
<td>$3T_{e-m} + 2T_{e-a} + 2T_h = 3.5238ms$</td>
</tr>
<tr>
<td>Zhao et al. [23]</td>
<td>$T_{e-m}+2T_h = 1.2ms$</td>
<td>$4T_{e-m}+3T_{e-a}+T_h = 4.6497ms$</td>
</tr>
<tr>
<td>Malhi et al. [24]</td>
<td>$T_{bp-m}+2T_{bp-a}$ + $T_h = 6.8862ms$</td>
<td>$3T_{bp}+3T_{bp-m}+T_{bp-a}$ + $2T_h = 17.8391ms$</td>
</tr>
<tr>
<td>Our scheme</td>
<td>$T_{e-m}+T_h=1.164ms$</td>
<td>$2T_{e-m}+3T_{e-a}+3T_h = 3.5577ms$</td>
</tr>
</tbody>
</table>
our scheme. In Figures 4(a)–4(c), we can get the conclusion that our scheme has a considerably low delay in sign and verification procedure, which reveals that our scheme has a much higher efficiency.

7. Conclusion

Since real application scenarios of VANETs require high efficiency, an efficient certificateless-based anonymous authentication and aggregate signature scheme are proposed. The proposed CLS and its improved scheme CL-AS are appropriate for VANETs due to analysis and testing. In addition, there is still some work to do in the future such as the low efficiency caused by the illegitimate signature in the aggregate verification process.

Data Availability

The proposed algorithm and its comparison rely on theoretical analysis. No additional test data sets are required in this paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Key International Cooperation Projects of the National Natural Science Foundation of China (No. 61520106007).

References


