


# An efficient certificateless aggregate signature scheme for the Internet of Vehicles

Yanan Zhao<sup>1</sup> | Yingzhe Hou<sup>1</sup> | Lili Wang<sup>1</sup> | Saru Kumari<sup>2</sup>  |  
 Muhammad Khurram Khan<sup>3</sup> | Hu Xiong<sup>1</sup>

<sup>1</sup>School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China

<sup>2</sup>Department of Mathematics, Chaudhary Charan Singh University, Meerut, India

<sup>3</sup>Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia

## Correspondence

Saru Kumari, Department of Mathematics, Chaudhary Charan Singh University, Meerut-250 001, India.  
 Email: saryusiirahi@gmail.com

Hu Xiong, School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China.  
 Email: xionghu.uestc@gmail.com

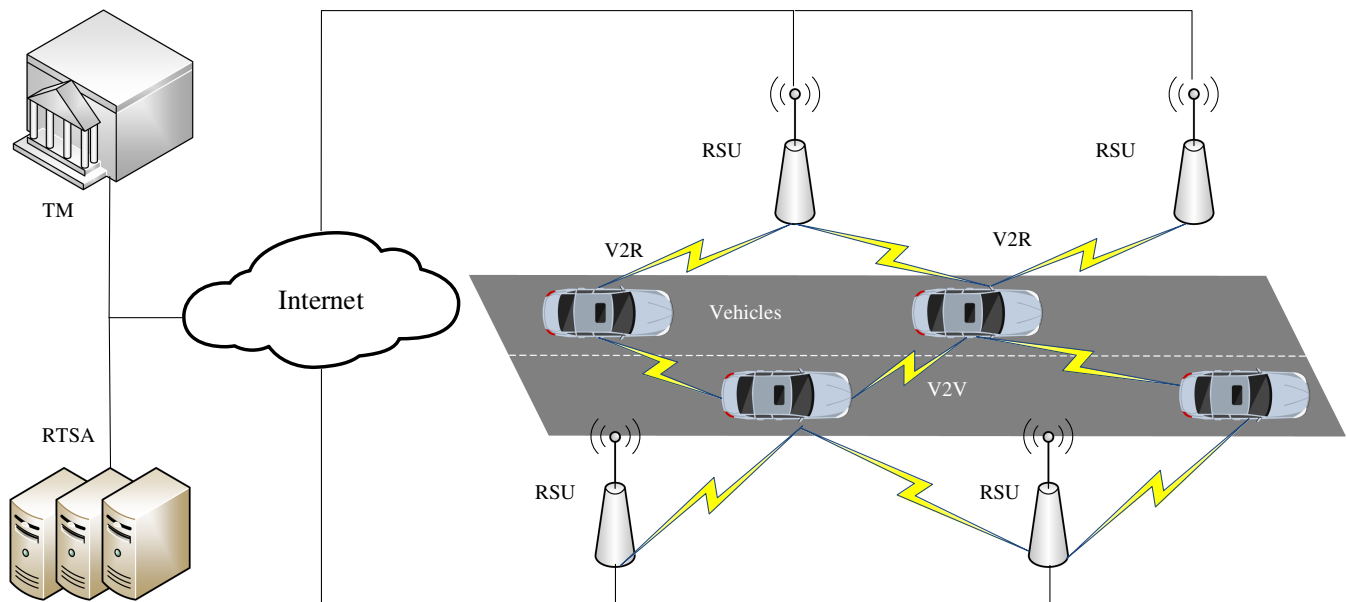
## Abstract

In recent years, the research of Internet of vehicles (IoV) has received extensive attention. In IoV, vehicles can make intelligent decisions by exchanging the real-time traffic information between other vehicles and IoV infrastructures, thereby reducing the probability of traffic jams and accidents. Although IoV has many advantages, it is necessary to ensure that the data are not edited, forged, or disclosed during the transmission. Also, the certificateless signature (CLS) seems to be the solution to this problem. While in the high-intensity data transmission environment, the CLS scheme is inefficient for application. To improve the efficiency, the function of aggregation was introduced into the CLS. The certificateless aggregate signature (CL-AS) can aggregate multiple signatures into a brief signature, reducing the computational cost and the size of signature. Nevertheless, most of the recent proposed CL-AS schemes have problems of security or efficiency. In this paper, we presented an advanced efficient CL-AS scheme with elliptic curve cryptography for the IoV environment. In addition, the proposed scheme uses pseudonyms in communications to prevent vehicles from revealing their identity. We proved the security of our proposal in the random oracle based on computational Diffie-Hellman assumption. Also, the efficiency analysis and simulation show the superiority of our scheme.

## 1 | INTRODUCTION

With the explosive growth of communication technology, fifth-generation (5G) networks have become a hot topic for the past few years.<sup>1</sup> In addition, 5G networks achieve high-speed data transmission while making it possible for everything to be interconnected.<sup>2</sup> As an essential utilization of 5G networks, the Internet of Vehicles (IoV) can transmit data among vehicles, roadside units (RSUs), data centers and personal devices with high speed. Also, a typical IoV scenario is illustrated in Figure 1. Compared with the traditional vehicular ad hoc networks (VANETs), IoV can make timely decisions to achieve intelligent driving assistance by analyzing the data gathered from multiple sensors and other vehicles. The probability of traffic jams and accidents can be reduced, thus decreasing travel time and pollution.<sup>3</sup>

Although IoV has many advantages, there are many obstacles to its widespread application. One of the obstacles is the privacy breaches. Without proper privacy protection, a malicious adversary can collect the information of vehicles, such as routes or status, to execute an attack. Fortunately, the application of pseudonyms in the communications can avoid this problem. Then, the vehicle can communicate with each other or with RSUs by using the pseudonym, and no one can get the true identity of the vehicle except a regional transport supervision agency (RTSA) and a trusted mechanism (TM). Even if the messages between the vehicles and the RSUs are collected by hackers, it will not reveal privacy.



**FIGURE 1** A typical Internet of vehicles scenario. RSU, roadside unit; RTSA, regional transport supervision agency; TM, trusted mechanism; V2R, vehicle-to-RSU; V2V, vehicle-to-vehicle

In addition, to prevent a malicious vehicle from escaping responsibility in case of dispute, the RTSA is able to reveal the true identity of the vehicle from the transmitting message. Another one is the security problem, such as the forgery attack. If the authenticity of the information in the IoV cannot be guaranteed, a malicious adversary can edit, forge, or inject data to perform attacks, which would result in a security catastrophe. To cope with this problem, all the data transmitted through the IoV should be signed.<sup>4</sup> While in conditions of high-intensity traffic, each vehicle or RSU needs to verify numerous signatures from other vehicles around it, which results in high verification cost. To reduce the computational cost, Boneh et al advanced the idea of the aggregate signature.<sup>5</sup> In their scheme, an aggregator aggregates  $n$  signatures of  $n$  messages from  $n$  users into a single short signature, and a verifier can confirm that  $n$  users sign the corresponding messages. As a result, the total signature size and the verification consumption are reduced. In the traditional aggregate signature schemes,<sup>6-8</sup> the user needs the trusted certificate authority (CA) to generate a public key certificate as a link connecting the user's identity and the corresponding public key. Nevertheless, a series of certificate management problems, such as the issuance and storage of certificates, have caused huge overhead. Then, to facilitate the management process of certificates, Cheng et al<sup>9</sup> presented the identity-based (ID-based) aggregate scheme. The ID-based aggregate scheme's superiority is to use a unique identification to identifies the user's identity as the user's public key, while the responsibility for generating the corresponding private key belongs to a third party called private key generator (PKG). However, the PKG can forge anyone's signatures easily, resulting in the key escrow problem. Then, a large amount of certificateless aggregate signature (CL-AS) schemes was introduced.<sup>10-14</sup>

In the CL-AS scheme, the user's private key is computed by combining the secret value chosen by the user and the user partial private key generated by a semi-trusted key generate center (KGC). By this means, the CL-AS scheme not only can settle the key escrow problem but also the certificate management problem.

Recently, a CL-AS scheme was constructed by Kamil and Ogundoyin,<sup>15</sup> and they claimed that the proposed scheme is secure. However, we found that Kamil and Ogundoyin's scheme cannot resist the type I adversary  $\mathcal{A}_1$ 's and the type II adversary  $\mathcal{A}_2$ 's attacks. To overcome the weakness, we propose a novel and efficient CL-AS scheme. The result of the proposed scheme's performance analysis demonstrates the superiority of our scheme in efficiency and security.

- First, this paper demonstrates that Kamil and Ogundoyin's scheme<sup>15</sup> is not safe enough to defend against the attacks from  $\mathcal{A}_1$  and  $\mathcal{A}_2$  adversaries.
- Next, an improved CL-AS scheme with elliptic curve cryptography (ECC) is proposed, which is proven secure in the random oracle model under the computational Diffie-Hellman assumption. Also, it can resist the attacks from  $\mathcal{A}_1$  and  $\mathcal{A}_2$  adversaries.
- Finally, the efficiency analysis and simulation show the superiority of our scheme in efficiency and practicality.

The rest of this paper is organized as follows. Section 2 shows the related works. Section 3 gives some preliminaries, including some notations, elliptic curve equation, our complexity assumptions, the notion and security model of CL-AS schemes. In Section 4, we analyze the Kamil and Ogundoyin's scheme. Our efficient scheme is described in Section 5. In Section 6, we compare our scheme with the other schemes. Finally, the conclusion is present in Section 7.

## 2 | RELATED WORK

### 2.1 | Certificateless public key cryptography (CL-PKC)

The conception of certificateless public key cryptography (CL-PKC) was first introduced by Al-Riyami and Paterson.<sup>16</sup> In their scheme, an undisclosed value chosen by user is combined with a partial private key produced by the KGC to generate the private key. By this way, not only the management of certificates in the PKI is simplified but also the identity-based cryptosystem's key escrow problem is solved. However, Huang et al<sup>17</sup> demonstrated that Al-Riyami and Paterson's scheme<sup>16</sup> can neither resist the attack from  $\mathcal{A}_1$  adversary nor the attack from  $\mathcal{A}_1$  adversary. Yum and Lee presented the first generic CLS scheme construction detail,<sup>18</sup> while Hu et al<sup>19</sup> pointed out the construction of Yum and Lee's scheme<sup>18</sup> is vulnerable against key replacement attack. Then, the first concrete CLS scheme in the standard model was advanced by Liu et al.<sup>20</sup> Unfortunately, the scheme of Liu et al failed to defend the malicious but passive KGC attack, which was demonstrated by Xiong et al.<sup>21</sup> Then, Xiong et al proposed a modified scheme,<sup>21</sup> while Xia et al<sup>22</sup> pointed out that the scheme of Xiong et al cannot resist the key replacement attacks. After that, plenty of CL-PKC schemes were presented successively.<sup>23-26</sup> However, most of the CL-PKC schemes proposed recently are insecure.

### 2.2 | Certificateless aggregate signature

Boneh et al first presented the concept of aggregate signature scheme.<sup>5</sup> Through the aggregation operation,  $n$  signatures, which are signed by  $n$  users on  $n$  messages, are aggregated into one brief signature. The aggregate signature can be verified to prove that  $n$  users sign on  $n$  messages. This scheme reduces the amount of signatures and the size of data transferred.

Combining the advantages of both the aggregate signature scheme and CL-PKC, the CL-AS was first advanced by Gong et al.<sup>27</sup> They gave two CL-AS schemes from bilinear pairing, respectively, while their schemes<sup>27</sup> require a clock synchronization to aggregate the signature, causing the performance degradation. After that, Xiong et al<sup>14</sup> advanced an efficient CL-AS scheme, which does not need clock synchronization. Furthermore, the scheme of Xiong et al uses a very small constant number of pairing computations, and only three pairing operations were required in verification, which is independent of the aggregated signatures' numbers. However, many researchers pointed out that the scheme of Xiong et al is vulnerable to the forgery attack.<sup>28-30</sup> Then, Horng et al<sup>12</sup> introduced a CL-AS scheme for VANETs, and they asserted that their proposal is secure against the adaptive chosen-message attack. However, Gayathri et al<sup>31</sup> claimed that the scheme of Horng et al<sup>12</sup> cannot resist the malicious-but-passive KGC attacks. Kumar et al<sup>32</sup> presented a CL-AS scheme with bilinear pairing. While on account of the heavy computational cost of bilinear pairing, much attention has been paid to the research of the CL-AS scheme without bilinear pairings to improve the performance.<sup>10,15,33,34</sup> Cui et al<sup>10</sup> advanced a CL-AS scheme with ECC. However, Kamil and Ogundoyin<sup>15</sup> declared that the scheme of Cui et al is not secure against the signature forgery attack and constructed a modified high efficiency signature scheme for VANETs, which does not use bilinear pairings. They claimed that their proposed scheme can meet all the requirements of VANETs about security and privacy. Unfortunately, we found the scheme of Kamil and Ogundoyin<sup>15</sup> to be not resistant to the forgery attack.

## 3 | PRELIMINARIES

### 3.1 | Elliptic curve cryptography

Let  $\mathbb{F}_q$  denote a prime finite field with prime order  $q$  and  $\mathbb{E}_q$  denote an elliptic curve.  $\mathbb{E}_q/\mathbb{F}_q$  demonstrates an elliptic curve  $\mathbb{E}_q$  over  $\mathbb{F}_q$ , which is defined by the following equation:  $y^2 = x^3 + ax + \beta \bmod q$ , where  $\alpha, \beta \in \mathbb{F}_q$  and  $(4\alpha^3 + 27\beta^2) \bmod q \neq 0$ .

Let the notation  $O$  denote the point at infinity. The additive group  $\mathbb{G}$  of the elliptic curve includes points over  $\mathbb{E}_q/\mathbb{F}_q$  and  $O$  under the computation of point addition  $T = U + V$  for  $U, V \in \mathbb{G}$  defined on the basis of a chord-and-tangent rule.

Suppose  $P$  is a generator of the group  $\mathbb{G}$ , and the order of  $\mathbb{G}$  is  $q$ . Let  $x \in \mathbb{Z}_q^*$ . We can define the scalar multiplication by the following equation:

$$xP = \underbrace{(P + P + \dots + P)}_{x \text{ times}}.$$

### 3.2 | Discrete logarithm problem

**Discrete logarithm problem:** There are two points  $P_1$  and  $P_2$  over  $\mathbb{E}_q/\mathbb{F}_q$ , which is the elliptical curve. Given  $P_2 = yP_1$  to compute  $y$ .

**Discrete logarithm problem assumption:** If there is no  $\hat{t}$ -time algorithm to figure out the discrete logarithm problem (DLP) with a probability of at least  $\varepsilon$ , then the  $(\hat{t}, \varepsilon)$ -DLP assumption holds.

### 3.3 | Certificateless aggregate signature scheme

Generally, a CL-AS scheme includes nine algorithms as follows: **Setup**, **RegistVehicle**, **GeneratePartialPrivateKey**, **GeneratePseudonym**, **GenerateVehicleKey**, **Sign**, **Verify**, **Aggregate**, and **AggregateVerify**.

1. **Setup:** The TM executes this probabilistic algorithm, which takes a security parameter  $z$  as input to generate a master secret key  $s$ , the master public key  $P_{\text{pub}}$ , and system parameters  $params$ .
2. **RegistVehicle:** The RTSA executes this algorithm. It takes vehicle's identity  $ID_\lambda$  as input to generate the hash chain set  $\tilde{r}_{y,\lambda}$ .
3. **GeneratePseudonym:** The RTSA executes this algorithm. It takes vehicle's identity  $ID_\lambda$  as input to generate pseudonym  $PID_{y,\lambda}$ .
4. **GeneratePartialPrivateKey:** The KGC executes this algorithm. It takes master secret key  $s$ , vehicle's pseudonym  $PID_{y,\lambda}$  as input. Then, the partial private key  $PSK_\lambda$  is generated, which should be transmitted to a vehicle  $\lambda$  via a secure communication channel.
5. **GenerateVehicleKey:** A vehicle  $\lambda$  executes this probabilistic algorithm. It takes the secret value  $x_\lambda$ , pseudoidentity  $PID_{y,\lambda}$ , partial private key  $PSK_\lambda$  as input to generate private key  $SK_\lambda$ , and public key  $PK_\lambda$ .
6. **Sign:** A vehicle  $\lambda$  executes this algorithm. It takes the vehicle's secret value  $x_\lambda$ , partial private key  $PSK_\lambda$ , vehicle's pseudonym  $PID_{y,\lambda}$ , and the message  $m_\lambda$  as input. Then, it generates a signature  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$  of the message  $m_\lambda$ .
7. **Verify:** The verifier executes this algorithm to verify a single signature. If this signature is valid, it outputs 1; otherwise, it outputs 0.
8. **Aggregate:** The aggregator executes this probabilistic algorithm. The algorithm inputs  $n$  vehicles' signatures  $(\sigma_1, \sigma_2, \dots, \sigma_n)$  on  $n$  distinct messages  $(m_1, m_2, \dots, m_n)$ . Then, the algorithm outputs the aggregate signature  $\sigma_n^*$  on  $(m_1, m_2, \dots, m_n)$ .
9. **AggregateVerify:** The verifier executes this algorithm. It inputs  $n$  vehicles' public keys  $(PK_1, PK_2, \dots, PK_n)$ , the vehicles' pseudonyms  $(PID_{y,1}, PID_{y,2}, \dots, PID_{y,n})$ , and the corresponding aggregate signature  $\sigma_n^*$  on the message set  $(m_1, m_2, \dots, m_n)$ . Finally, if this signature is valid, the output is 1; otherwise, the output is 0.

### 3.4 | Security model

In this section, we consider two different types of adversaries:  $\mathcal{A}_1$  and  $\mathcal{A}_2$ .  $\mathcal{A}_1$  can execute public key replacement attacks without knowing  $s$ , while  $\mathcal{A}_2$  can access  $s$ , but does not allow public key replacement attacks to be performed.

An adversary  $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$  and a challenger  $\mathcal{B}$  simulate the security of CLS or CL-AS by interacting with two games.  $\mathcal{A}$  can execute the five oracles as follows.

1. **RevealPublicKey:** Given a vehicle's pseudoidentity  $PID_{y,\lambda}$ , the oracle returns  $PK_\lambda$  to  $\mathcal{A}$ .
2. **RevealPartialPrivateKey:** Given a vehicle's pseudoidentity  $PID_{y,\lambda}$ , the oracle returns partial private key  $PSK_\lambda$  corresponding to the vehicle to  $\mathcal{A}$ .
3. **RevealSecretValue:** Given a pseudoidentity  $PID_{y,\lambda}$  of a vehicle, the oracle returns the secret value  $x_\lambda$  of this vehicle to  $\mathcal{A}$ .
4. **ReplacePublicKey:** Given a pseudoidentity  $PID_{y,\lambda}$  and  $PK_\lambda$  of a vehicle, the oracle replaces the corresponding public key  $PK_\lambda$  with  $PK_\lambda^*$ .
5. **Sign:** Given a message  $m_\lambda \in \{0, 1\}^*$ , the oracle generates a signature  $\sigma_\lambda$  corresponding to the  $m_\lambda$  and returns it to  $\mathcal{A}$ .

In CLS, we define the following two games.

**Definition 1.** If there exist no polynomial time  $\mathcal{A}_1$  to win **Game I** with nonnegligible advantage, the CLS scheme will be proved secure in adaptive chosen-message attacks.

**Game I.** A challenger  $B$  and  $\mathcal{A}_1$  play this game as follows:

*Setup:*  $B$  executes the algorithm of **Setup** to output  $s$ ,  $params$ , and  $P_{pub}$ . Finally,  $B$  returns  $params$  to  $\mathcal{A}_1$  and retains  $s$  for itself.

*Query:*  $\mathcal{A}_1$  can query the oracles of *RevealPublicKey*, *RevealPartialPrivateKey*, *RevealSecretValue*, *ReplacePublicKey*, and *Sign*.

*Forgery:*  $B$  returns a signature  $\sigma_\lambda^*$  on  $m_\lambda^* \in \{0, 1\}^*$ , which is produced by a vehicle with pseudonym  $PID_{y,\lambda}^*$  and the corresponding public key  $PK_\lambda^*$  to  $\mathcal{A}_1$ .

$\mathcal{A}_1$  wins **Game I** if the following steps are satisfied:

1. The oracle *RevealPartialPrivateKey* has never been queried with  $PID_{y,\lambda}^*$ .
2. The signature  $\sigma_\lambda^*$  generated by the corresponding message  $m_\lambda^*$  is valid.
3.  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been as the input of the oracle *Sign* to query its corresponding signature.

**Definition 2.** If there exist no polynomial time  $\mathcal{A}_2$  to win **Game II** with nonnegligible advantage, the CLS scheme will be proved secure in adaptive chosen-message attacks.

**Game II.** A challenger  $B$  and  $\mathcal{A}_2$  play this game as follows.

*Setup:*  $B$  executes the algorithm of **Setup** to output  $s$ ,  $params$ , and  $P_{pub}$ . Finally,  $B$  returns  $params$  and  $s$  to  $\mathcal{A}_2$ .

*Query:*  $\mathcal{A}_2$  can query the oracles of *RevealPublicKey*, *RevealSecretValue*, and *Sign*.

*Forgery:*  $B$  returns a signature  $\sigma_\lambda^*$  on  $m_\lambda^* \in \{0, 1\}^*$ , which is produced by a vehicle with pseudonym  $PID_{y,\lambda}^*$  and the corresponding public key  $PK_\lambda^*$  to  $\mathcal{A}_2$ .

$\mathcal{A}_2$  wins **Game II** if the following steps are satisfied:

1. The oracle *RevealSecretValue* has never been queried with  $PID_{y,\lambda}^*$ .
2. The signature  $\sigma_\lambda^*$  generated by the corresponding message  $m_\lambda^*$  is valid.
3.  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been as the input of the oracle *Sign* to query its corresponding signature.

In CL-AS, we define the following two games:

**Definition 3.** If there exist no polynomial time  $\mathcal{A}_1$  to win **Game III** with nonnegligible advantage, the CL-AS scheme will be proved secure in adaptive chosen-message attacks.

**Game III.** The players in this game are similar to **Game I**.

*Setup:* This step is similar to that shown in **Game I** above.

*Query:*  $\mathcal{A}_1$  can query the oracles of *RevealPublicKey*, *RevealPartialPrivateKey*, *RevealSecretValue*, *ReplacePublicKey*, and *Sign*.

*Forgery:*  $\mathcal{A}_1$  can forged a corresponding aggregate signature  $\sigma^*$  on a group of messages  $\{m_1^*, m_2^*, \dots, m_n^*\}$  from vehicles.

$\mathcal{A}_1$  wins **Game III** if the following steps are satisfied:

1. At least one pseudonym is not used as the oracle *RevealPartialPrivateKey*'s input to query  $PSK_\lambda$ .
2. The aggregate signature  $\sigma^*$  is a valid.
3.  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been as the input of the oracle *Sign* to query its corresponding signature.

**Definition 4.** If there exist no polynomial time  $\mathcal{A}_2$  to win **Game IV** with nonnegligible advantage, the CL-AS scheme will be proved secure in adaptive chosen-message attacks.

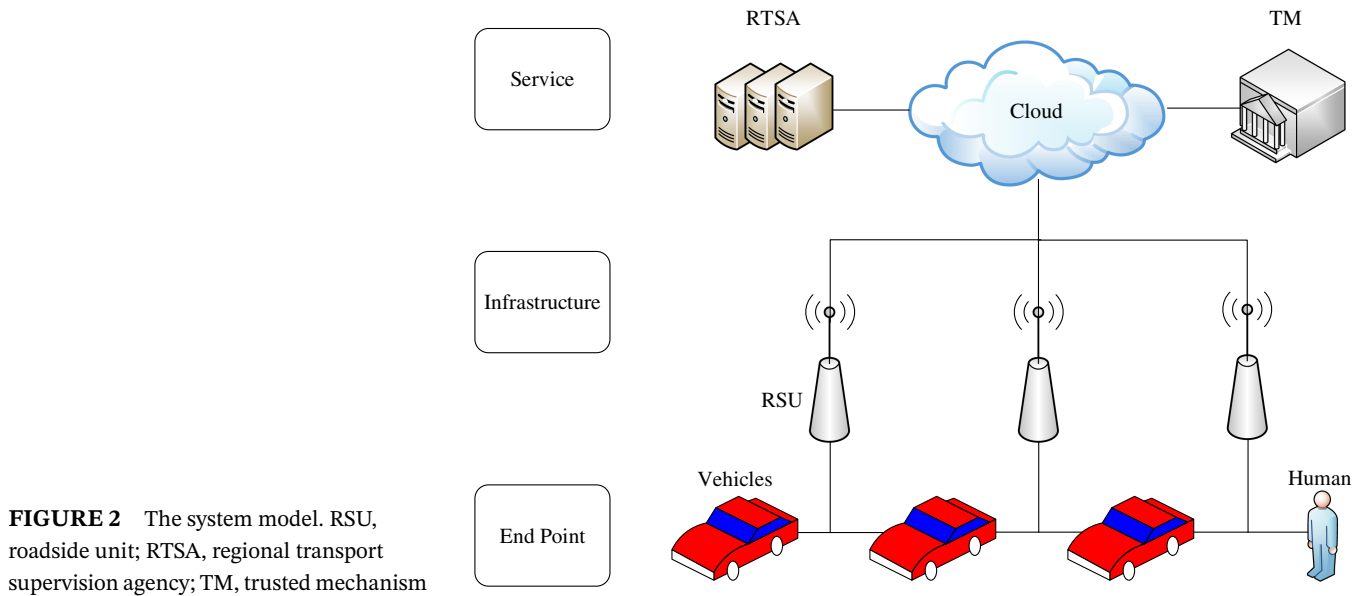
**Game IV.** The players in this game are similar to **Game II**.

*Setup:* This step is similar to that shown in **Game II** above.

*Query:*  $\mathcal{A}_2$  can query the oracles of *RevealPublicKey*, *RevealSecretValue*, and *Sign*.

*Forgery:*  $\mathcal{A}_2$  can forgery a corresponding aggregate signature  $\sigma^*$  on a group of messages  $\{m_1^*, m_2^*, \dots, m_n^*\}$  from vehicles.





**FIGURE 2** The system model. RSU, roadside unit; RTSA, regional transport supervision agency; TM, trusted mechanism

$\mathcal{A}_2$  wins **Game IV** if the following steps are satisfied:

1. At least one pseudonym is not used as the oracle *RevealSecretValue's* or *ReplacePublicKey's* input to query  $x_\lambda$ .
2. The aggregate signature  $\sigma^*$  is valid.
3.  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been the input of the oracle *Sign* to query its corresponding signature.

### 3.5 | System model

In order to clearly describe the proposed system, the system model is shown in Figure 2. In our scheme, there are four participants: a TM, an RTSA, an RSU, and a vehicle. Due to the different communication methods, the system can be divided to two levels. The upper level involves wired communication among TM, RTSA, and RSU, while the lower level involves wireless communication of 5G network between vehicle and RSU. The description of each participant is as follows:

1. TM: It is a trusted third-party responsible for vehicle registration, system initialization, and parameter generation. As an important part of our scheme, it can track malicious vehicles, providing conditional traceability. TM has enough storage and processing capabilities.
2. RTMA (regional transport supervision agency): It is a trusted third-party responsible for partial private key generation. Similar to TM, regional transport supervision agency can also provide conditional traceability and has enough storage and processing capabilities.
3. RSU: It is a communication device installed along the road side, which is wired connected to TM and RTMA. RSUs and vehicles are connected by 5G network. RSU is responsible for collecting, uploading, and distributing traffic information. Comparing to TM and RTMA, RSU has limited storage and processing capabilities.
4. Vehicle: It is equipped with an on-board unit, which can transfer traffic information such as location information, vehicle speed, and vehicle direction with RSUs and other vehicles. Vehicles have limited battery, storage, and processing capabilities.

### 3.6 | Security requirements

There are several security properties that a practical scheme must achieve, including the following:

1. Authentication: A verifier RSU must be certain that a received message is from a legitimate user and its content has not been maliciously altered.
2. Anonymity: The pseudonym mechanism is used by vehicles to communicate with other vehicles so that privacy can be reflected and the disclosure of identity information can be avoided.
3. Unlinkability: Any two or more messages should not be linked to the same signer by the attacker, and any two signatures should not be linked to the same signer by an adversary, malicious vehicle, or RSU.

4. Traceability: The TM tracks the driving records of all vehicles by using the corresponding pseudonyms to master the evidence and avoid judgment errors.
5. Resistance to attacks: The proposed scheme must be able to guard resist some well-known attacks such as replay and modification.

## 4 | A BRIEF ANALYSIS

### 4.1 | Overview of Kamil and Ogundoyin's CLS scheme

The symbols used in this article are shown in Table 1. In this way, symbolic narration is reduced.

1. **Setup:** Given a security parameter  $z$ , the TM outputs the system parameters  $params$  by executing the algorithms as follows.
  - Two secure prime numbers  $p$  and  $q$  are selected at random. Then, randomly pick  $a, b \in \mathbb{F}_q$  and equation  $y^2 = x^3 + ax + b \pmod{p}$  that defined the elliptic curve  $\mathbb{E}$ . Finally, select a generator  $P$  of the additive group  $\mathbb{G}$ , which is composed of all points on  $\mathbb{E}_q$ .
  - Randomly select the master secret key  $s \in \mathbb{Z}_q^*$  and compute  $P_{pub} = s \cdot P$  as the system public key.
  - Randomly choose five secure hash functions:  $h_0 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ ,  $h_1 : \{0, 1\}^* \times \mathbb{G}^2 \rightarrow \mathbb{Z}_q^*$ ,  $h_2 : \{0, 1\}^* \times \mathbb{G}^3 \rightarrow \mathbb{Z}_q^*$ ,  $h_3 : \{0, 1\}^* \times \mathbb{G}^4 \rightarrow \mathbb{Z}_q^*$ ,  $h_4 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G} \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ .
  - The current time is  $t_i$ , and then select the time function  $f(t_i)$  as a tool to confirm the current timeslot.
  - Keep the  $s$  secret and output the public parameters

$$params = \{p, q, a, b, P, P_{pub}, h_0, h_1, h_2, h_3, h_4, f(t_i)\}.$$

2. **RegistVehicle:** After receiving an identify  $ID_\lambda$  from vehicle, the RTSA registers the vehicle by performing the following operations.

Notation	Description
TM	Trusted mechanism
RTSA	Regional transport supervision agency
KGC	Key generator center
AS	Application server
RSU	Roadside unit
OBU	On-board unit
$ID_\lambda$	True identity of entity $\lambda$
$h_\lambda(\cdot), \lambda = 0, 1, 2, 3, 4, 5$	Hash function
$p, q$	Prime numbers
$F_q$	A prime finite field
$E_q/F_q$	An elliptic curve $E_q$ on $F_q$
$\mathbb{G}$	A group of cyclic additive
$P$	Generator of the group $\mathbb{G}$
$Z_q$	Minimum residual set of modulo $q$
$t_s$	Timeslot
$T_\lambda$	The entity $\lambda$ 's timestamp
$PSK_\lambda$	The entity $\lambda$ 's partial private key
$PK_\lambda$	The entity $\lambda$ 's public key
$x_\lambda$	The system's master secret key
$P_{pub}$	The system's public key
$k$	The RTSA's secret key
$SK_\lambda$	The entity $\lambda$ 's private key
$  $	The operation of concatenation
$\Delta T$	The delay of transmission
$PID_{y,\lambda}$	The entity $\lambda$ 's pseudonym in timeslot $y$
$m_\lambda$	The entity $\lambda$ 's message

**TABLE 1** List of notations

- Randomly select  $\tilde{t}_{1,\lambda} \in \mathbb{Z}_q^*$  and calculate the hash chain  $\tilde{t}_{y,\lambda} = h_0(\tilde{t}_{y-1,\lambda})$ . For instance,  $\tilde{t}_{2,\lambda} = h_0(\tilde{t}_{1,\lambda})$ ,  $\tilde{t}_{3,\lambda} = h_0(\tilde{t}_{2,\lambda})$ , ...
- Calculate the hash chain set

$$\tilde{t}_{y,\lambda} = \{\tilde{t}_{2,\lambda}, \tilde{t}_{3,\lambda}, \dots, \tilde{t}_{n,\lambda}\}.$$

3. **GeneratePartialPrivateKey:** After receiving *params* and a vehicle with identity  $ID_\lambda$ , the KGC generates the corresponding partial private key  $PSK_\lambda$  as follows.

- Randomly pick  $k \in \mathbb{Z}_q^*$ .
- Choose  $\nabla$  as the state information and  $ID_{RTSA}$  as the identity of RTSA.
- Calculate its public key  $PK_{RTSA} = k \cdot P$ . Then, compute  $\alpha_\lambda = h_2(P_{pub} \| ID_\lambda)$ ,  $\beta_\lambda = h_2(ID_\lambda \| ID_{RTSA} \| k \| \nabla)$ ,  $t = h_2(\nabla)$ , and  $\xi_\lambda = h_2(ID_\lambda \| PK_{RTSA})$ . Finally, calculate  $A_\lambda = t\beta_\lambda \cdot P$  and  $k_\lambda = t\beta_\lambda + \xi_\lambda \alpha_\lambda k$ .
- Output  $PSK_\lambda = (A_\lambda, k_\lambda)$  and publish the  $PK_{RTSA}$ .
- Return  $\{PSK_\lambda = (A_\lambda, k_\lambda), \tilde{t}_{y,\lambda}\}$  to a vehicle and return  $(\tilde{t}_{1,\lambda}, ID_\lambda)$  to TM.

4. **GeneratePseudonym:** After receiving the tuple  $\{(A_\lambda, k_\lambda), \tilde{t}_{y,\lambda}\}$  from the KGC, the RTSA executes the following operations for producing a vehicle's time-variant pseudonym set.

- Check if the following equation holds or not:

$$k_\lambda \cdot P \stackrel{?}{=} A_\lambda + \xi_\lambda \cdot \alpha_\lambda \cdot PK_{RTSA}$$

If so,  $PSK_\lambda$  is valid.

- Obtain  $ts_{cur}$  as the current timeslot via  $f(t_i)$ .
- Calculate  $PID_{y,\lambda} = h_1(ID_\lambda \| \tilde{t}_{y,\lambda} \| \nabla \| ts_{cur})$ . Finally, output the pseudonym set  $\{PID_{1,\lambda}, PID_{2,\lambda}, \dots, PID_{n,\lambda}\}$ .

5. **GenerateVehicleKey:** After receiving *params*, a vehicle with pseudoidentity  $PID_{y,\lambda}$  produces its private key  $SK_\lambda$  and public key  $PK_\lambda$ , respectively.

- Randomly pick  $a_\lambda, r_\lambda^1, r_\lambda^2 \in \mathbb{Z}_q^*$ .
- Calculate  $SK_\lambda^1 = h_3(r_\lambda^1 \| A_\lambda \| PID_{y,\lambda})$ , and  $SK_\lambda^2 = h_3(r_\lambda^2 \| k_\lambda \| PID_{y,\lambda})$ .
- Output  $SK_\lambda = a_\lambda(SK_\lambda^1 + SK_\lambda^2)$  and  $PK_\lambda = SK_\lambda \cdot P$ .

6. **Sign:** After receiving *params*,  $PSK_\lambda$ ,  $SK_\lambda$ , and  $PK_\lambda$ , a vehicle with pseudoidentity  $PID_{y,\lambda}$  can sign on a message  $m_\lambda$  as follows.

- Randomly pick  $d_\lambda \in \mathbb{Z}_q^*$  and calculate  $w_\lambda = \xi_\lambda \alpha_\lambda$ .
- Choose  $T_\lambda$  as a present timestamp chosen by entity  $\lambda$ .
- Calculate  $v_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| SK_\lambda^1 \| SK_\lambda^2 \| T_\lambda)$ ,  $h_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| w_\lambda \| PK_\lambda \| PK_{RTSA} \| P_{pub} \| T_\lambda)$ , and  $\delta_\lambda = h_4(m_\lambda \| PK_\lambda \| \nabla \| T_\lambda)$ .
- Calculate  $y_\lambda = d_\lambda v_\lambda$ ,  $\Omega_\lambda = y_\lambda \cdot P$ ,  $R_\lambda = \delta_\lambda \cdot PK_\lambda + h_\lambda \cdot A_\lambda + \Omega_\lambda$ , and  $\phi_\lambda = \delta_\lambda SK_\lambda + h_\lambda k_\lambda + d_\lambda v_\lambda$ .

Finally, the signer outputs a signature  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$  that corresponds to the message  $m_\lambda$  and transmits  $(PID_{y,\lambda}, m_\lambda, PK_\lambda, w_\lambda, \sigma_\lambda, T_\lambda)$  to the verifier.

7. **Verify:** After receiving the tuple  $(PID_{y,\lambda}, m_\lambda, PK_\lambda, w_\lambda, \sigma_\lambda, T_\lambda)$ , the algorithm can be executed by any verifier to verify the signature as follows.

- Choose  $\Delta T$  as the allowable network transmission delay.  $T_\lambda$  is the send time of signature and  $T_2$  is the receive time of signature.
- Check if the following equation holds:  $T_2 - T_\lambda \leq \Delta T$ . If it holds,  $T_\lambda$  is valid and receives the signature; otherwise, it aborts.
- Calculate  $h_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| w_\lambda \| PK_\lambda \| PK_{RTSA} \| P_{pub} \| T_\lambda)$ .
- Check if the following equation

$$\phi_\lambda \cdot P \stackrel{?}{=} R_\lambda + h_\lambda w_\lambda \cdot PK_{RTSA}$$

holds or not. If so, it accepts the signature; otherwise, it discards the signature.



## 4.2 | Overview of Kamil and Ogundoyin CL-AS scheme

The **Setup**, **RegistUser**, **GeneratePartialPrivateKey**, **GeneratePseudonym**, **GenerateVehicleKey**, **Sign**, **Verify** algorithms, and the CLS above are the same. Furthermore, the **Aggregate** and **AggregateVerify** are demonstrated as follows.

1. **Aggregate**: The generator of aggregate signature such as the RSU can do this algorithm. When RSU obtains a group of message signature pairs  $\{(m_1, t_1, \sigma_1 = (R_1, \phi_1)), (m_2, t_2, \sigma_2 = (R_2, \phi_2)), \dots, (m_n, t_n, \sigma_n = (R_n, \phi_n))\}$  from  $n$  vehicles  $\{V_1, V_2, \dots, V_n\}$ , it computes  $R = \sum_{\lambda=1}^n R_\lambda$  and  $\phi = \sum_{\lambda=1}^n \phi_\lambda$ . Finally, The RSU produces a CL-AS  $\sigma = (R, \phi)$ .
2. **AggregateVerify**: This algorithm is performed by application server (AS) or RSU. After receiving the CL-AS  $\sigma = (R, \phi)$ , the AS or RSU checks whether the timestamp  $T_\lambda$  is satisfying the requirement, where  $\lambda = 1, 2, \dots, n$ . If it holds, it executes the following operations; otherwise, it aborts.
  - Compute  $h_\lambda = h_4(PID_{y,\lambda} || m_\lambda || w_\lambda || PK_\lambda || PK_{RTSA} || P_{pub} || T_\lambda)$ .
  - Check whether

$$\phi \cdot P = R + \sum_{\lambda=1}^n h_\lambda w_\lambda \cdot PK_{RTSA}$$

holds or not. If it holds, it receives the signature; otherwise, the signature is rejected.

## 4.3 | Forgery attack for the CLS of Kamil and Ogundoyin

Kamil and Ogundoyin claimed that their scheme<sup>15</sup> was secure and existentially unforgeable against  $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ . Unfortunately, it is not the truth. We found that the security of scheme by Kamil and Ogundoyin<sup>15</sup> is too easy to be broken and gave a specific attack example. Any adversary has the ability to execute the steps as follows.

1. Pick  $r^* \in \mathbb{Z}_q^*$  at random; let  $\phi_\lambda^* = r^*$ .
2. After receiving *params*, a message  $m_\lambda$ ,  $P_{pub}$ ,  $PID_{y,\lambda}$ ,  $PK_\lambda$ , and  $PK_{RTSA}$ ,  $\mathcal{A}$  can compute  $w_\lambda = \xi_\lambda \alpha_\lambda$  and  $h_\lambda = h_4(PID_{y,\lambda} || m_\lambda || w_\lambda || PK_\lambda || PK_{RTSA} || P_{pub} || T_\lambda)$ .
3. After receiving the  $h_\lambda$  and  $w_\lambda$ ,  $\mathcal{A}$  can compute

$$R_\lambda^* = \phi_\lambda^* \cdot P - h_\lambda w_\lambda \cdot PK_{RTSA}.$$

4. The signature  $\sigma_\lambda^* = (R_\lambda^*, \phi_\lambda^*)$  corresponding to any message  $m_\lambda^*$  can be generated by  $\mathcal{A}$ .

From the previously mentioned analysis, we received a conclusion that  $\sigma_\lambda^*$  is a valid signature. In addition, the validity of a signature can be verified by any vehicle and involves the following specific operation.

- Check whether the equation  $\phi_\lambda^* \cdot P \stackrel{?}{=} R_\lambda^* + h_\lambda w_\lambda \cdot PK_{RTSA}$  holds or not.

Based on the following analysis, we can receive that this verification is holding certainly:

$$\begin{aligned} \phi_\lambda^* \cdot P &= (\phi_\lambda^* \cdot P - h_\lambda w_\lambda \cdot PK_{RTSA}) + h_\lambda w_\lambda \cdot PK_{RTSA} \\ &= R_\lambda^* + h_\lambda w_\lambda \cdot PK_{RTSA}. \end{aligned}$$

Therefore, we can easily forge a signature successfully without knowing  $s$  or without replacing a vehicle's  $PK_\lambda$ . The root cause of fatal flaws in the work of Kamil and Ogundoyin<sup>15</sup> is that the system public key  $P_{pub}$  and a vehicle's public key  $PK_\lambda$  are not embedded in CLS's **Verify** algorithm.

## 4.4 | Forgery attack for the CL-AS of Kamil and Ogundoyin

1. Randomly select  $\{r_1^*, r_2^*, \dots, r_n^*\} \in \mathbb{Z}_q^*$ , and calculate  $\phi^* = \sum_{i=1}^n r_i^*$ , where  $1 \leq i \leq n$ .
2. After receiving *params*, a message  $m_i$ ,  $P_{pub}$ ,  $PID_{y,i}$ , and the RTSA's public key  $PK_{RTSA}$ ,  $\mathcal{A}$  can compute  $w_i = \xi_i \alpha_i$  and  $h_i = h_4(PID_{y,i} || m_i || w_i || PK_i || PK_{RTSA} || P_{pub} || T_i)$ .

3. After receiving the  $h_i$  and  $w_i$ ,  $\mathcal{A}$  can compute

$$R^* = \sum_{i=1}^n r_i^* \cdot P - \sum_{i=1}^n h_i w_i \cdot PK_{RTSA}.$$

4. A set of  $n$  forged message-signature pairs  $\{(m_i^*, \sigma_i^* = (R_i^*, \phi_1^*)), i = 1, \dots, n\}$  corresponding to any message  $m_i^*$  can be generated by  $\mathcal{A}$ .

From the previously mentioned analysis, we received a conclusion that  $\sigma_i^*$  ( $i = 1, \dots, n$ ) is a valid signature. In addition, the validity of a signature can be verified by any vehicle and involves the following specific operation.

- Check whether the equation  $\phi^* \cdot P \stackrel{?}{=} R^* + \sum_{i=1}^n h_i w_i \cdot PK_{RTSA}$  holds or not.

The correctness of  $\sigma_i^*$  is verified as follows:

$$\phi^* \cdot P = \left( \sum_{i=1}^n r_i^* \cdot P - \sum_{i=1}^n h_i w_i \cdot PK_{RTSA} \right) + \sum_{i=1}^n h_i w_i \cdot PK_{RTSA}$$

Thus, the CL-AS scheme of Kamil and Ogundoyin<sup>15</sup> cannot resist either the attack from  $\mathcal{A}_1$  or the attack from  $\mathcal{A}_2$ .

## 5 | OUR PROPOSED SCHEME

### 5.1 | Our CLS scheme

1. **Setup:** Given a security parameter  $z$ , the TM outputs the system parameters *params* by executing the algorithms as follows.
  - Two secure prime numbers  $p$  and  $q$  are selected at random. Then, randomly pick  $a, b \in \mathbb{F}_q$ , and equation  $y^2 = x^3 + ax + b \pmod{p}$  that defined the elliptic curve  $\mathbb{E}_q$ . Finally, select a generator  $P$  of the additive group  $\mathbb{G}$ , which is composed of all points on  $\mathbb{E}_q$ .
  - Randomly select the master secret key  $s \in \mathbb{Z}_q^*$  and compute  $P_{\text{pub}} = s \cdot P$  as the system public key.
  - Randomly select six secure hash functions:  $h_0 : \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$ ,  $h_1 : \{0, 1\}^* \times \mathbb{Z}_q^* \times \{0, 1\}^* \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$ ,  $h_2 : \{0, 1\}^* \times \{0, 1\}^* \times \mathbb{G}^3 \rightarrow \mathbb{Z}_q^*$ ,  $h_3 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ ,  $h_4 : \mathbb{Z}_q^* \times \{0, 1\}^* \times \mathbb{Z}_q^* \times \mathbb{G}^3 \times \mathbb{Z}_q^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$ , and  $h_5 : \{0, 1\}^* \times \mathbb{G} \times \{0, 1\}^* \times \mathbb{Z}_q^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$ .
  - The current time is  $t_i$  and then select the time function  $f(t_i)$  as a tool to confirm the current timeslot.
  - Keep the  $s$  secret and output the public parameters *params*:

$$\{p, q, a, b, P, P_{\text{pub}}, h_0, h_1, h_2, h_3, h_4, h_5, f(t_i)\}.$$

2. **RegistVehicle:** After receiving an identify  $ID_\lambda$  from a vehicle, the RTSA registers the vehicle by performing the following operations.

- Randomly select  $\tilde{t}_{1,\lambda} \in \mathbb{Z}_q^*$ , and calculate the hash chain  $\tilde{t}_{y,\lambda} = h_0(\tilde{t}_{y-1,\lambda})$ . For instance,  $\tilde{t}_{2,\lambda} = h_0(\tilde{t}_{1,\lambda})$ ,  $\tilde{t}_{3,\lambda} = h_0(\tilde{t}_{2,\lambda})$ ,  $\dots$ .
- Calculate the hash chain set

$$\tilde{t}_{y,\lambda} = \{\tilde{t}_{2,\lambda}, \tilde{t}_{3,\lambda}, \dots, \tilde{t}_{n,\lambda}\}.$$

3. **GeneratePseudonym:** After receiving an identity from a vehicle, the RTSA executes the following operations for producing a vehicle's time-variant pseudonym set.

- Obtain  $ts_{\text{cur}}$  as the current timeslot via  $f(t_i)$ .
- Calculate  $PID_{y,\lambda} = h_1(ID_\lambda \| \tilde{t}_{y,\lambda} \| \nabla \| ts_{\text{cur}})$ . Finally, output the pseudonym set  $\{PID_{1,\lambda}, PID_{2,\lambda}, \dots, PID_{n,\lambda}\}$ .

4. **GeneratePartialPrivateKey:** After receiving a vehicle with pseudonym  $PID_{y,\lambda}$ , the KGC generates the corresponding partial private key  $PSK_\lambda$  as follows.

- Randomly pick  $k, r_\lambda, \beta_\lambda, \xi_\lambda \in \mathbb{Z}_q^*$ .
  - Calculate  $R_\lambda = r_\lambda \cdot P$  and  $PK_{\text{RTSA}} = k \cdot P$ .
  - Calculate  $\alpha_\lambda = h_2(PID_{y,\lambda} \| P_{\text{pub}} \| PK_{\text{RTSA}} \| R_\lambda)$ ,  $t_\lambda = h_3(\nabla)$ ,  $\xi_\lambda = t_\lambda \cdot \beta_\lambda$ .
  - Calculate  $A_\lambda = \xi_\lambda P$ ,  $s_\lambda = \xi_\lambda + \alpha_\lambda s$ .
  - Generate  $PSK_\lambda = (A_\lambda, s_\lambda)$ , and publish the  $PK_{\text{RTSA}}$ .
  - Return  $\{PSK_\lambda = (A_\lambda, s_\lambda), \tilde{t}_{y,\lambda}\}$  to the vehicle.
5. **GenerateVehicleKey:** After receiving partial private key  $PSK_\lambda$ , a vehicle with pseudoidentity  $PID_{y,\lambda}$  outputs its private key  $SK_\lambda$  and public key  $PK_\lambda$ , respectively.
- Check whether the equation:

$$s_\lambda \cdot P \stackrel{?}{=} A_\lambda + \alpha_\lambda \cdot P_{\text{pub}}$$

holds or not. If so,  $PSK_\lambda$  is valid.

- Randomly pick  $x_\lambda \in \mathbb{Z}_q^*$  as the secret value of a vehicle.
  - Let  $SK_\lambda = \{PSK_\lambda, x_\lambda\}$ .
  - Calculate  $PK_\lambda = x_\lambda \cdot P$ .
6. **Sign:** After receiving  $PSK_\lambda$ ,  $x_\lambda$ , and  $PK_\lambda$ , a vehicle with pseudoidentity  $PID_{y,\lambda}$  can sign on a message  $m_\lambda$  as follows:
- Calculate  $h_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| \alpha_\lambda \| PK_\lambda \| PK_{\text{RTSA}} \| P_{\text{pub}} \| T_\lambda \| R_\lambda)$  and  $\delta_\lambda = h_5(m_\lambda \| PK_\lambda \| \nabla \| T_\lambda \| R_\lambda)$ .
  - Calculate  $\phi_\lambda = r_\lambda + h_\lambda(\delta_\lambda x_\lambda + s_\lambda)$ .

Finally, the signer outputs a signature  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$  corresponding to the message  $m_\lambda$  and transmits  $(PID_{y,\lambda}, m_\lambda, PK_\lambda, w_\lambda, \sigma_\lambda, T_\lambda)$  to the verifier.

7. **Verify:** After receiving the tuple  $(PID_{y,\lambda}, m_\lambda, PK_\lambda, \alpha_\lambda, \sigma_\lambda, T_\lambda)$ , the algorithm can be executed by any verifier to verify the signature as follows.
- Choose  $\Delta T$  as the allowable network transmission delay.  $T_\lambda$  is the send time of signature, and  $T_2$  is the receive time of signature.
  - Check if the following equation holds:  $T_2 - T_\lambda \leq \Delta T$ . If it holds,  $T_\lambda$  is valid and receives the signature; otherwise, it aborts.
  - Calculate  $h_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| \alpha_\lambda \| PK_\lambda \| PK_{\text{RTSA}} \| P_{\text{pub}} \| T_\lambda \| R_\lambda)$  and  $\delta_\lambda = h_5(m_\lambda \| PK_\lambda \| \nabla \| T_\lambda \| R_\lambda)$ .
  - Check whether the following equation

$$\phi_\lambda \cdot P \stackrel{?}{=} R_\lambda + h_\lambda(\delta_\lambda PK_\lambda + A_\lambda + \alpha_\lambda P_{\text{pub}}).$$

holds or not. If so, it accepts the signature; otherwise, it discards the signature.

## 5.2 | Our CL-AS scheme

The **Setup**, **RegistUser**, **GeneratePartialPrivateKey**, **GeneratePseudonym**, **GenerateVehicleKey**, **Sign**, and **Verify** algorithms of CL-AS are similar to the CLS scheme. Furthermore, the **Aggregate** and **AggregateVerify** are demonstrated as follows.

1. **Aggregate:** The generator of aggregate signature such as the RSU can do this algorithm. When RSU obtains a group of message signature pairs  $\{(m_1, t_1, \sigma_1 = (R_1, \phi_1)), (m_2, t_2, \sigma_2 = (R_2, \phi_2)), \dots, (m_n, t_n, \sigma_n = (R_n, \phi_n))\}$  from  $n$  vehicles  $\{V_1, V_2, \dots, V_n\}$ , it computes  $\phi = \sum_{\lambda=1}^n \phi_\lambda$ . Finally, The RSU produces an aggregate certificateless signature  $\sigma = (R_1, R_2, \dots, R_n, \phi)$ .
2. **AggregateVerify:** This algorithm is performed by AS or RSU. After receiving the CL-AS  $\sigma = (R_1, R_2, \dots, R_n, \phi)$ , the AS or RSU checks whether the timestamp  $T_\lambda$  is satisfying the requirement, where  $\lambda = 1, 2, \dots, n$ . If it holds, it executes the following operations; otherwise, it aborts.
  - Compute  $h_\lambda = h_4(PID_{y,\lambda} \| m_\lambda \| \alpha_\lambda \| PK_\lambda \| PK_{\text{RTSA}} \| P_{\text{pub}} \| T_\lambda \| R_\lambda)$ ,  $\delta_\lambda = h_5(m_\lambda \| PK_\lambda \| \nabla \| T_\lambda \| R_\lambda)$ .

- Check whether the equation

$$\phi \cdot P = \sum_{\lambda=1}^n R_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} \delta_{\lambda} PK_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} A_{\lambda} + \left( \sum_{\lambda=1}^n h_{\lambda} \alpha_{\lambda} \right) \cdot P_{\text{pub}}$$

holds. If it holds, it receives the signature; otherwise, the signature is rejected.

### 5.3 | Correctness of single message verification

We can prove the correctness of our scheme by executing the following operations:

$$\begin{aligned} \phi_{\lambda} \cdot P &= (r_{\lambda} + h_{\lambda}(\delta_{\lambda} x_{\lambda} + s_{\lambda}))P \\ &= r_{\lambda}P + h_{\lambda}(\delta_{\lambda} x_{\lambda}P + \xi_{\lambda}P + \alpha_{\lambda}sP) \\ &= R_{\lambda} + h_{\lambda}(\delta_{\lambda} PK_{\lambda} + A_{\lambda} + \alpha_{\lambda}P_{\text{pub}}) \end{aligned}$$

### 5.4 | Correctness of aggregate verification

After receiving  $\sigma = (R_1, R_2, \dots, R_n, \phi)$  on  $n$  messages, the AS or RSU checks whether  $\phi \cdot P = \sum_{\lambda=1}^n R_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} \delta_{\lambda} PK_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} A_{\lambda} + (\sum_{\lambda=1}^n h_{\lambda} \alpha_{\lambda})P_{\text{pub}}$  holds. The following operation is the proof for correctness:

$$\begin{aligned} \phi \cdot P &= \left( \sum_{\lambda=1}^n \phi_{\lambda} \right) \cdot P \\ &= \left( \sum_{\lambda=1}^n r_{\lambda} + h_{\lambda}(\delta_{\lambda} x_{\lambda} + s_{\lambda}) \right) \cdot P \\ &= \left( \sum_{\lambda=1}^n r_{\lambda} + h_{\lambda}(\delta_{\lambda} x_{\lambda} + \xi_{\lambda} + \alpha_{\lambda}s) \right) \cdot P \\ &= \left( \sum_{\lambda=1}^n r_{\lambda} \cdot P \right) + \left( \sum_{\lambda=1}^n h_{\lambda} \delta_{\lambda} x_{\lambda} \cdot P \right) + \left( \sum_{\lambda=1}^n h_{\lambda} \xi_{\lambda} \cdot P \right) + \left( \sum_{\lambda=1}^n h_{\lambda} \alpha_{\lambda} s \cdot P \right) \\ &= \sum_{\lambda=1}^n R_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} \delta_{\lambda} PK_{\lambda} + \sum_{\lambda=1}^n h_{\lambda} A_{\lambda} + \left( \sum_{\lambda=1}^n h_{\lambda} \alpha_{\lambda} \right) P_{\text{pub}}. \end{aligned}$$

### 5.5 | Security proof of the proposed CLS scheme

We demonstrate that our CLS is provably security in the random oracle model, where  $h_i$  ( $i = 0, \dots, 5$ ) as six random oracles and the following theorems are shown.

**Theorem 1.** *The introduced CLS is existentially unforgeable against  $\mathcal{A}_1$  in the random oracle model.*

*Proof.* Given a DLP example  $Q = s \cdot P$ ,  $\mathcal{A}_1$  has the ability to execute the attack of public key replacement. We play the **Game I** between  $\mathcal{A}_1$  and a challenger  $\mathcal{B}$ , which can solve the DLP with nonnegligible probability  $\Pr[\text{Succ}(\mathcal{A}_1)]$ . The standard for  $\mathcal{B}$  to win this game is to generate  $s$  through interacting with  $\mathcal{A}_1$ .

*Setup:*  $\mathcal{B}$  randomly selects  $PID^*$  as the challenged identity during **Game I**. Subsequently,  $\mathcal{B}$  returns  $params = \{p, q, a, b, P, P_{\text{pub}}, h_0, h_1, h_2, h_3, h_4, h_5, f(t_i)\}$  to  $\mathcal{A}_1$ . It picks  $s \in \mathbb{Z}_q^*$  at random and computes  $P_{\text{pub}} = s \cdot P$ .  $\mathcal{B}$  maintains seven hash lists  $L_{h_0}, L_{h_1}, L_{h_2}, L_{h_3}, L_{h_4}, L_{h_5}$  and  $L_k$ , while  $\mathcal{A}_1$  has the ability to perform oracle queries.

*$h_0$  Queries:* When  $\mathcal{A}_1$  queries the oracle of  $h_0$ ,  $\mathcal{B}$  searches  $L_{h_0}$  for the tuple  $(m_{\lambda}, \tilde{y}_{y,\lambda})$ . If it can be found,  $\mathcal{B}$  submits  $\tilde{y}_{y,\lambda}$  to  $\mathcal{A}_1$ ; otherwise, it randomly picks  $\tilde{y}_{y,\lambda} \in \mathbb{Z}_q^*$  and adds  $(m_{\lambda}, \tilde{y}_{y,\lambda})$  to  $L_{h_0}$ . Eventually,  $\mathcal{B}$  submits  $\tilde{y}_{y,\lambda}$  to  $\mathcal{A}_1$ .

*$h_1$  Queries:* When  $\mathcal{A}_1$  queries the oracle of  $h_1$ ,  $\mathcal{B}$  searches  $L_{h_1}$  for the tuple  $(ID_{\lambda}, \tilde{y}_{y,\lambda}, \nabla, ts_{\text{cur}}, PID_{y,\lambda})$ . If it can be found,  $\mathcal{B}$  submits  $PID_{y,\lambda}$  to  $\mathcal{A}_1$ ; otherwise, it selects  $PID_{y,\lambda} \in \mathbb{Z}_q^*$  at random and adds them to  $L_{h_1}$ . Eventually,  $\mathcal{B}$  returns  $PID_{y,\lambda}$  to  $\mathcal{A}_1$ .

*$h_2$  Queries:* When  $\mathcal{A}_1$  queries the oracle of  $h_2$ ,  $\mathcal{B}$  searches  $L_{h_2}$  for the tuple  $(PID_{y,\lambda}, P_{\text{pub}}, PK_{\text{RTSA}}, R_{\lambda}, \alpha_{\lambda})$ . If it can be found,  $\mathcal{B}$  submits  $\alpha_{\lambda}$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  picks  $\alpha_{\lambda} \in \mathbb{Z}_q^*$  at random and adds it to  $L_{h_2}$ . Eventually,  $\mathcal{B}$  submits  $\alpha_{\lambda}$  to  $\mathcal{A}_1$ .

$h_3$  Queries: When  $\mathcal{A}_1$  queries the oracle of  $h_3$ ,  $\mathcal{B}$  searches  $L_{h_3}$  for the tuple  $(\nabla, t_\lambda)$ . If it can be found,  $\mathcal{B}$  submits  $t_\lambda$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  picks  $t_\lambda \in \mathbb{Z}_q^*$  at random and adds them to  $L_{h_3}$ . Eventually,  $\mathcal{B}$  submits  $t_\lambda$  to  $\mathcal{A}_1$ .

$h_4$  Queries: When  $\mathcal{A}_1$  queries the oracle of  $h_4$ ,  $\mathcal{B}$  randomly searches  $L_{h_4}$  for the tuple  $(PID_{y,\lambda}, m_\lambda, \alpha_\lambda, PK_\lambda, PK_{RTSA}, P_{pub}, T_\lambda, R_\lambda, h_\lambda)$ . If it can be found,  $\mathcal{B}$  submits  $h_\lambda$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  picks  $h_\lambda \in \mathbb{Z}_q^*$  at random and adds them to  $L_{h_4}$ . Eventually,  $\mathcal{B}$  submits  $h_\lambda$  to  $\mathcal{A}_1$ .

$h_5$  Queries: When  $\mathcal{A}_1$  queries the oracle of  $h_5$ ,  $\mathcal{B}$  searches  $L_{h_5}$  for the tuple  $(m_\lambda, PK_\lambda, \nabla, T_\lambda, R_\lambda, \delta_\lambda)$ . If it can be found,  $\mathcal{B}$  submits  $\delta_\lambda$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  picks  $\delta_\lambda \in \mathbb{Z}_q^*$  at random and adds them to  $L_{h_5}$ . Eventually,  $\mathcal{B}$  submits  $\delta_\lambda$  to  $\mathcal{A}_1$ .

*RevealPublicKey*: When  $\mathcal{A}_1$  queries the oracle,  $\mathcal{B}$  performs the following steps:

- If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  randomly selects  $\xi_\lambda, b_\lambda, x_\lambda \in \mathbb{Z}_q^*$ , sets  $s_\lambda = \xi_\lambda, \alpha_\lambda = b_\lambda, A_\lambda = \xi_\lambda P - b_\lambda P_{pub}, PK_\lambda = x_\lambda P$ .  $\mathcal{B}$  adds  $(PID_{y,\lambda}, P_{pub}, PK_{RTSA}, R_\lambda, \alpha_\lambda)$  to  $L_{h_1}$  and adds  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$  to  $L_k$ . The  $s_\lambda, \alpha_\lambda, A_\lambda$  produced in this way must satisfy the following equation:  $s_\lambda P = A_\lambda + \alpha_\lambda P_{pub}$ . Finally,  $\mathcal{B}$  returns  $PK_\lambda$  to  $\mathcal{A}_1$ .
- If  $PID_{y,\lambda} = PID^*$ ,  $\mathcal{B}$  randomly selects  $\xi_\lambda, b_\lambda, x_\lambda \in \mathbb{Z}_q^*$ , and sets  $s_\lambda = \perp, \alpha_\lambda = b_\lambda, A_\lambda = \xi_\lambda P, PK_\lambda = x_\lambda P$ .  $\mathcal{B}$  adds  $(PID_{y,\lambda}, P_{pub}, PK_{RTSA}, R_\lambda, \alpha_\lambda)$  to  $L_{h_1}$  and adds  $\langle PID_{y,\lambda}, \perp, A_\lambda, x_\lambda, PK_\lambda \rangle$  to  $L_k$ . Finally,  $\mathcal{B}$  returns  $PK_\lambda$  to  $\mathcal{A}_1$ .

*RevealPartialPrivateKey*: When  $\mathcal{A}_1$  takes  $PID_{y,\lambda}$  as input to query this oracle,  $\mathcal{B}$  performs the following steps:

- If  $PID_{y,\lambda} = PID^*$ , it aborts.
- If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  searches the list  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ . If it can be found,  $\mathcal{B}$  submits  $s_\lambda$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  executes a *RevealPublicKey* query on  $PID_{y,\lambda}$  and submits  $s_\lambda$  to  $\mathcal{A}_1$ .

*RevealSecretValue*:  $\mathcal{B}$  searches the list  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ . If it can be found,  $\mathcal{B}$  submits  $x_\lambda$  to  $\mathcal{A}_1$ ; otherwise,  $\mathcal{B}$  executes a *RevealPublicKey* query on  $PID_{y,\lambda}$  and submits  $x_\lambda$  to  $\mathcal{A}_1$ .

*ReplacePublicKey*: When  $\mathcal{A}_1$  takes  $(PID_{y,\lambda}, m_\lambda)$  as input to execute this query, if the list  $L_k$  has  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ , sets  $PK_\lambda = PK_\lambda^*$  and  $x_\lambda = \perp$ ; otherwise,  $\mathcal{B}$  executes a *RevealPublicKey* query on  $PID_{y,\lambda}$  and sets  $PK_\lambda = PK_\lambda^*, x_\lambda = \perp$ .

*Sign*: When  $\mathcal{A}_1$  takes  $(PID_{y,\lambda}, m_\lambda)$  as input to execute this query,  $\mathcal{B}$  randomly selects  $a_\lambda, b_\lambda \in \mathbb{Z}_q^*$  and sets  $\phi_\lambda = a_\lambda, h_\lambda = b_\lambda, R_\lambda = a_\lambda P - b_\lambda(\delta_\lambda PK_\lambda + A_\lambda + \alpha_\lambda P_{pub})$ .  $\mathcal{B}$  returns  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$  to  $\mathcal{A}_1$  and adds  $\langle PID_{y,\lambda}, m_\lambda, \alpha_\lambda, PK_\lambda, PK_{RTSA}, P_{pub}, T_\lambda, R_\lambda, h_\lambda \rangle$  to  $L_{h_4}$ . The  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$  generated in this way must satisfy the following equation:

$$\phi_\lambda P = R_\lambda + h_\lambda(\delta_\lambda PK_\lambda + A_\lambda + \alpha_\lambda P_{pub})$$

*Forgery*:  $\mathcal{A}_1$  outputs a tuple  $(PID_{y,\lambda}, m_\lambda, \sigma_\lambda)$  as its forgery, where  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$ . If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  aborts the simulation; otherwise,  $\mathcal{B}$  searches  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ .  $\mathcal{B}$  replays the above simulation process using the identical random tape without the same selection of the hash function  $h_i$  ( $i = 0, \dots, 5$ ), and  $\mathcal{A}_1$  will output signatures  $\sigma_\lambda^{(j)} = (R_\lambda, \phi_\lambda^{(j)})$ , where  $j = 1, \dots, 4$ . Due to those signatures are valid, the equation will hold as follows:

$$\phi_\lambda^{(j)} \cdot P = R_\lambda + h_\lambda^{(j)} \left( \delta_\lambda^{(j)} PK_\lambda + A_\lambda + \alpha_\lambda^{(j)} P_{pub} \right) \quad j = 1, 2, 3, 4.$$

Thus, we can obtain  $R_\lambda = r_\lambda \cdot P, PK_\lambda = x_\lambda \cdot P, A_\lambda = \xi_\lambda \cdot P$  and  $P_{pub} = s \cdot P$ . Finally, we could receive four equations by calculating the above equations as follows:

$$\phi_\lambda^{(j)} = r_\lambda + h_\lambda^{(j)} (\delta_\lambda^{(j)} x_\lambda + \xi_\lambda + \alpha_\lambda^{(j)} s) \quad j = 1, 2, 3, 4$$

Except  $r_\lambda, x_\lambda, \xi_\lambda$  and  $s$  are all known for  $\mathcal{B}$ .  $\mathcal{B}$  can get four unknown numbers from four equations. Thus, the value of  $s$  can be obtained and the solution of DLP problem will be cracked.

*Analysis*: We use three events to examine the success probability of DLP as follows. Suppose that the advantage of  $\epsilon$  was given in forging signatures by  $\mathcal{A}_1$  and it never got the similar input value from hash functions.

- $E_1$ :  $\mathcal{B}$  does not abandon *RevealPartialPrivateKey*'s query.
- $E_2$ :  $\mathcal{A}_1$  could forge a valid signature  $\langle PID_{y,\lambda}, m_\lambda, \sigma_\lambda \rangle$ .
- $E_3$ : The outputting tuple  $\langle PID_{y,\lambda}, m_\lambda, \sigma_\lambda \rangle$  satisfies  $PID_{y,\lambda} = PID^*$ .

We could obtain that  $\Pr[E_1] \geq \left(1 - \frac{1}{q_{h_1}}\right)^{q_{EPPK}}$ ,  $\Pr[E_2|E_1] \geq \epsilon$ , and  $\Pr[E_3|E_1 \wedge E_2] \geq \left(1 - \frac{1}{q_{h_1}}\right)$ , where  $q_{h_1}$  and  $q_{EPPK}$  denote the numbers of  $h_1$  queries and *RevealPartialPrivateKey* queries, respectively. So, we can solve the DLP with the

probability  $\Pr[\text{Succ}(\mathcal{A}_1)]$ .

$$\begin{aligned}\Pr[\text{Succ}(\mathcal{A}_1)] &= \Pr[E_1 \wedge E_2 \wedge E_3] \\ &= \Pr[E_1] \Pr[E_2|E_1] \Pr[E_3|E_1 \wedge E_2] \\ &\geq \frac{1}{q_{h_1}} \left(1 - \frac{1}{q_{h_1}}\right)^{q_{\text{EPPK}}} \epsilon.\end{aligned}$$

Through the above analysis, we can know that  $\mathcal{B}$  can solve DLP, which contradicts the hardness assumption of the DLP. Therefore, Theorem 1 can be verified.  $\square$

**Theorem 2.** *The introduced CLS is existentially unforgeable against  $\mathcal{A}_2$  in the random oracle model.*

*Proof.* Given a DLP example  $Q = x_\lambda \cdot P$ ,  $\mathcal{A}_2$  has the ability to execute the attack of malicious passive KGC. We play the **Game II** between  $\mathcal{A}_2$  and a challenger  $\mathcal{B}$ , which can solve the DLP with nonnegligible probability  $\Pr[\text{Succ}(\mathcal{A}_2)]$ . The standard for  $\mathcal{B}$  to win this game is to output  $x_\lambda$  through interacting with  $\mathcal{A}_2$ .

**Game II** is the same as **Game I** besides the imitations of *RevealPublicKey*, *RevealPartialPrivateKey*, and *RevealSecretValue* oracles. The three oracles are described as follows.

*RevealPublicKey*: When  $\mathcal{A}_2$  executes this query,  $\mathcal{B}$  can make the following operations.

- If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  randomly selects  $\beta_\lambda, x_\lambda \in \mathbb{Z}_q^*$ , computes  $\alpha_\lambda = h_2(PID_{y,\lambda} \| P_{\text{pub}} \| PK_{\text{RTSA}} \| R_\lambda)$ ,  $t_\lambda = h_2(\nabla)$ , and sets  $\xi_\lambda = t_\lambda \beta_\lambda$ ,  $A_\lambda = \xi_\lambda P$ ,  $s_\lambda = \xi_\lambda + \alpha_\lambda s$ ,  $PK_\lambda = x_\lambda P$ .  $\mathcal{B}$  adds  $(PID_{y,\lambda}, P_{\text{pub}}, PK_{\text{RTSA}}, R_\lambda, \alpha_\lambda)$  to  $L_{h_1}$ , adds  $(\nabla, t_\lambda)$  to  $L_{h_2}$ , and adds  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$  to  $L_k$ . Then,  $\mathcal{B}$  returns  $PK_\lambda$  to  $\mathcal{A}_2$  as a response.
- If  $PID_{y,\lambda} = PID^*$ ,  $\mathcal{B}$  randomly selects  $\beta_\lambda, x_\lambda \in \mathbb{Z}_q^*$ , computes  $\alpha_\lambda = h_2(PID_{y,\lambda} \| P_{\text{pub}} \| PK_{\text{RTSA}} \| R_\lambda)$ ,  $t_\lambda = h_2(\nabla)$ , and sets  $\xi_\lambda = t_\lambda \beta_\lambda$ ,  $A_\lambda = \xi_\lambda P$ ,  $s_\lambda = \xi_\lambda + \alpha_\lambda s$ ,  $x_\lambda = \perp$ ,  $PK_\lambda = x_\lambda P$ .  $\mathcal{B}$  adds  $(PID_{y,\lambda}, P_{\text{pub}}, PK_{\text{RTSA}}, R_\lambda, \alpha_\lambda)$  to  $L_{h_1}$ , adds  $(\nabla, t_\lambda)$  to  $L_{h_2}$ , and adds  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, \perp, PK_\lambda \rangle$  to  $L_k$ . Then,  $\mathcal{B}$  returns  $PK_\lambda$  to  $\mathcal{A}_2$  as a response.

*RevealPartialPrivateKey*: When  $\mathcal{A}_2$  executes this query,  $\mathcal{B}$  searches  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ . If it can be found,  $\mathcal{B}$  submits  $s_\lambda$  to  $\mathcal{A}_2$ ; otherwise,  $\mathcal{B}$  executes a *RevealPublicKey* query on  $PID_{y,\lambda}$  and submits  $s_\lambda$  to  $\mathcal{A}_2$ .

*RevealSecretValue*: When  $\mathcal{A}_2$  executes the query,  $\mathcal{B}$  can make the following operations:

- If  $PID_{y,\lambda} = PID^*$ , it aborts.
- If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  searches  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ . If it can be found,  $\mathcal{B}$  submits  $x_\lambda$  to  $\mathcal{A}_2$ . Otherwise,  $\mathcal{B}$  executes a *RevealPublicKey* query on  $PID_{y,\lambda}$  and submits  $x_\lambda$  to  $\mathcal{A}_2$ .

*Forgery*:  $\mathcal{A}_2$  outputs a tuple  $(PID_{y,\lambda}, m_\lambda, \sigma_\lambda)$  as its forgery, where  $\sigma_\lambda = (R_\lambda, \phi_\lambda)$ . If  $PID_{y,\lambda} \neq PID^*$ ,  $\mathcal{B}$  aborts the simulation; Otherwise,  $\mathcal{B}$  searches  $L_k$  for the tuple  $\langle PID_{y,\lambda}, s_\lambda, A_\lambda, x_\lambda, PK_\lambda \rangle$ .  $\mathcal{B}$  replays the above simulation process using the identical random tape without the same selection of the hash function  $h_i$  ( $i = 0, \dots, 5$ ),  $\mathcal{A}_2$  will output signatures  $\sigma_\lambda^{(j)} = (R_\lambda, \phi_\lambda^{(j)})$ , where  $j = 1, 2, 3$ . Due to those signatures are valid, the equation will hold as follows:

$$\phi_\lambda^{(j)} \cdot P = R_\lambda + h_\lambda^{(j)} \left( \delta_\lambda^{(j)} PK_\lambda + A_\lambda + \alpha_\lambda^{(j)} P_{\text{pub}} \right) \quad j = 1, 2, 3$$

Thus, we can obtain  $R_\lambda = r_\lambda \cdot P$ ,  $PK_\lambda = x_\lambda \cdot P$ ,  $A_\lambda = \xi_\lambda \cdot P$  and  $P_{\text{pub}} = s \cdot P$ . Finally, we can receive three equations by calculating the above equations as follows:

$$\phi_\lambda^{(j)} = r_\lambda + h_\lambda^{(j)} \left( \delta_\lambda^{(j)} x_\lambda + \xi_\lambda + \alpha_\lambda^{(j)} s \right) \quad j = 1, 2, 3$$

Except  $r_\lambda, x_\lambda, \xi_\lambda$  are all known for  $\mathcal{B}$ .  $\mathcal{B}$  can get three unknown numbers from three equations. Thus, the value of  $x_\lambda$  can be obtained and the solution of DLP problem will be cracked.

*Analysis*: We use three events to examine the success probability of DLP as follows. Suppose that the advantage of  $\epsilon$  was given in forging signatures by  $\mathcal{A}_2$ , and it never got the similar input value from hash functions.

- $E_1$ :  $\mathcal{B}$  does not abandon the *RevealSecretKey*'s query.
- $E_2$ :  $\mathcal{A}_2$  has the ability to forge a valid signature  $\langle PID_{y,\lambda}, m_\lambda, \sigma_\lambda \rangle$ .
- $E_3$ : The outputting tuple  $\langle PID_{y,\lambda}, m_\lambda, \sigma_\lambda \rangle$  satisfies  $PID_{y,\lambda} = PID^*$ .



We could obtain that  $\Pr[E_1] \geq \left(1 - \frac{1}{q_{h_1}}\right)^{q_{ESV}}$ ,  $\Pr[E_2|E_1] \geq \epsilon$ ,  $\Pr[E_3|E_1 \wedge E_2] \geq \left(1 - \frac{1}{q_{h_1}}\right)$ , where  $q_{h_1}$  and  $q_{ESV}$  denote the numbers of  $h_1$  queries and *RevealSecretValue* queries, respectively. So, we can solve the DLP with the probability  $\Pr[\text{Succ}(\mathcal{A}_2)]$ .

$$\begin{aligned}\Pr[\text{Succ}(\mathcal{A}_2)] &= \Pr[E_1 \wedge E_2 \wedge E_3] \\ &= \Pr[E_1] \Pr[E_2|E_1] \Pr[E_3|E_1 \wedge E_2] \\ &\geq \frac{1}{q_{h_1}} \left(1 - \frac{1}{q_{h_1}}\right)^{q_{ESV}} \epsilon.\end{aligned}$$

Through the above analysis, we can know that  $\mathcal{B}$  can solve DLP, which contradicts the hardness assumption of the DLP. Therefore, Theorem 2 can be verified.  $\square$

## 5.6 | Security proof of the proposed CL-AS scheme

We demonstrate that our CL-AS is provably security in the random oracle model, where  $h_i$  ( $i = 0, \dots, 5$ ) as six random oracles and the following theorems are shown.

**Theorem 3.** *The introduced CL-AS is existentially unforgeable against  $\mathcal{A}_1$  in the random oracle model.*

*Proof.* Given a DLP example  $Q = s \cdot P$ ,  $\mathcal{A}_1$  has the ability to execute the attack of public key replacement. We play the **Game III** between  $\mathcal{A}_1$  and a challenger  $\mathcal{B}$ , which can solve the DLP with nonnegligible probability. The standard for  $\mathcal{B}$  to win this game is to output  $s$  through interacting with  $\mathcal{A}_1$ .

*Setup:*  $\mathcal{B}$  randomly selects  $PID^*$  as the challenged identity during **Game III**. Subsequently,  $\mathcal{B}$  returns  $params = \{p, q, a, b, P, P_{pub}, h_0, h_1, h_2, h_3, h_4, h_5, f(t_i)\}$  to  $\mathcal{A}_1$ . It picks  $s \in \mathbb{Z}_q^*$  at random and computes  $P_{pub} = s \cdot P$ .  $\mathcal{B}$  maintains seven hash lists  $L_{h_0}, L_{h_1}, L_{h_2}, L_{h_3}, L_{h_4}, L_{h_5}$  and  $L_k$ , while  $\mathcal{A}_1$  has the ability to perform oracle queries.

$\mathcal{A}_1$  could simulate the  $h_i$  Queries, where  $i = 0, 1, \dots, 5$ , *RevealPublicKey*, *RevealPartialPrivateKey*, *RevealSecretValue*, *ReplacePublicKey*, and *Sign* queries described in CLS above.

*Forgery:*  $\mathcal{A}_1$  generates a forged aggregate signature  $\sigma = (R_1, R_2, \dots, R_n, \phi)$  on message-identity-public key pairs  $(m_\lambda, PID_{y,\lambda}, PK_\lambda)$ , where  $(1 \leq \lambda \leq n)$ . Nevertheless, the following steps must hold:

- $\sigma$  is a valid aggregate signature.

$$\phi \cdot P = \sum_{\lambda=1}^n R_\lambda + \sum_{\lambda=1}^n h_\lambda \delta_\lambda PK_\lambda + \sum_{\lambda=1}^n h_\lambda A_\lambda + \left( \sum_{\lambda=1}^n h_\lambda \alpha_\lambda \right) P_{pub}$$

- At least one pseudonym does not used as the oracle *RevealPartialPrivateKey*'s input to query  $PSK_\lambda$  and  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been as the input of the oracle *Sign* to query its corresponding signature.

$\mathcal{B}$  replays the above simulation process using the identical random tape without the same selection of the hash function  $h_i$  ( $i = 0, \dots, 5$ ),  $\mathcal{A}_1$  will output signatures  $\sigma_\lambda^{(j)} = (R_\lambda, \phi_\lambda^{(j)})$ , where  $j = 1, \dots, 4$ . Due to those signatures being valid, the equation will hold:

$$\phi^{(j)} \cdot P = \sum_{\lambda=1}^n R_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \delta_\lambda^{(j)} PK_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} A_\lambda + \left( \sum_{\lambda=1}^n h_\lambda^{(j)} \alpha_\lambda^{(j)} \right) P_{pub} \quad j = 1, 2, 3, 4$$

Thus, we can obtain  $R_\lambda = r_\lambda \cdot P$ ,  $PK_\lambda = x_\lambda \cdot P$ ,  $A_\lambda = \xi_\lambda \cdot P$  and  $P_{pub} = s \cdot P$ . Finally, we could receive four equations by calculating the above equations as follows:

$$\phi^{(j)} = \sum_{\lambda=1}^n r_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \delta_\lambda^{(j)} x_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \xi_\lambda + \left( \sum_{\lambda=1}^n h_\lambda^{(j)} \alpha_\lambda^{(j)} \right) s \quad j = 1, 2, 3, 4$$

Except  $r_\lambda, x_\lambda, \xi_\lambda$  and  $s$  are all known for  $\mathcal{B}$ .  $\mathcal{B}$  can get four unknown numbers from four equations. Thus, the value of  $s$  can be obtained and the solution of DLP problem will be cracked.

Through the above analysis, we can know that  $\mathcal{B}$  can solve DLP, which contradicts the hardness assumption of the DLP. Therefore, Theorem 3 can be verified.  $\square$

**Theorem 4.** *The introduced CL-AS is existentially unforgeable against  $\mathcal{A}_2$  in the random oracle model.*

*Proof.* Given a DLP example  $Q = x_\lambda \cdot P$ ,  $\mathcal{A}_2$  has the ability to execute the attack of malicious but passive KGC. We play the **Game IV** between  $\mathcal{A}_2$  and a challenger  $\mathcal{B}$ , which can solve the DLP with nonnegligible probability. The standard for  $\mathcal{B}$  to win this game is to output  $x_\lambda$  through interacting with  $\mathcal{A}_2$ .

*Setup:*  $\mathcal{B}$  randomly selects  $PID^*$  as the challenged identity during **Game IV**. Subsequently,  $\mathcal{B}$  returns  $params = \{p, q, a, b, P, P_{pub}, h_0, h_1, h_2, h_3, h_4, h_5, f(t_i)\}$  to  $\mathcal{A}_2$ . It picks  $s \in \mathbb{Z}_q^*$  at random and computes  $P_{pub} = s \cdot P$ .  $\mathcal{B}$  maintains seven hash lists  $L_{h_0}, L_{h_1}, L_{h_2}, L_{h_3}, L_{h_4}, L_{h_5}$  and  $L_k$ , while  $\mathcal{A}_2$  has the ability to perform oracle queries.

$\mathcal{A}_2$  could simulate the  $h_i$  Queries,  $i = 0, 1, \dots, 5$ , *RevealPublicKey*, *RevealPartialPrivateKey*, *RevealSecretValue*, *ReplacePublicKey*, and *Sign* queries described in CLS above.

*Forgery:*  $\mathcal{A}_2$  generates a forged aggregate signature  $\sigma = (R_1, R_2, \dots, R_n, \phi)$  on message-identity-public key pairs  $(m_\lambda, PID_{y,\lambda}, PK_\lambda)$  with  $(1 \leq \lambda \leq n)$ . Nevertheless, the following steps must hold:

- $\sigma$  is a valid aggregate signature.

$$\phi \cdot P = \sum_{\lambda=1}^n R_\lambda + \sum_{\lambda=1}^n h_\lambda \delta_\lambda PK_\lambda + \sum_{\lambda=1}^n h_\lambda A_\lambda + \left( \sum_{\lambda=1}^n h_\lambda \alpha_\lambda \right) P_{pub}$$

- At least one pseudonym does not used as the oracle *RevealPartialPrivateKey*'s input to query  $PSK_\lambda$  and  $(PID_{y,\lambda}^*, m_\lambda^*)$  has never been as the input of the oracle *Sign* to query its corresponding signature.

$\mathcal{B}$  replays the above simulation process using the identical random tape without the same selection of the hash function  $h_i$  ( $i = 0, \dots, 5$ ),  $\mathcal{A}_2$  will output signatures  $\sigma_\lambda^{(j)} = (R_\lambda, \phi_\lambda^{(j)})$ , where  $j = 1, 2, 3$ . Due to those signatures are valid, the equation will hold:

$$\phi^{(j)} \cdot P = \sum_{\lambda=1}^n R_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \delta_\lambda^{(j)} PK_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} A_\lambda + \left( \sum_{\lambda=1}^n h_\lambda^{(j)} \alpha_\lambda^{(j)} \right) P_{pub} \quad j = 1, 2, 3$$

Thus, we can obtain  $R_\lambda = r_\lambda \cdot P$ ,  $PK_\lambda = x_\lambda \cdot P$ ,  $A_\lambda = \xi_\lambda \cdot P$  and  $P_{pub} = s \cdot P$ . Finally, we could receive three equations by calculating the above equations as follows:

$$\phi^{(j)} = \sum_{\lambda=1}^n r_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \delta_\lambda^{(j)} x_\lambda + \sum_{\lambda=1}^n h_\lambda^{(j)} \xi_\lambda + \left( \sum_{\lambda=1}^n h_\lambda^{(j)} \alpha_\lambda^{(j)} \right) s \quad j = 1, 2, 3$$

Except  $r_\lambda, x_\lambda, \xi_\lambda$  are all known for  $\mathcal{B}$ .  $\mathcal{B}$  can get three unknown numbers from three equations. Thus, the value of  $x_\lambda$  can be obtained and the solution of DLP problem will be cracked.

Through the above analysis, we can know that  $\mathcal{B}$  can solve DLP, which contradicts the hardness assumption of the DLP. Therefore, Theorem 4 can be verified.  $\square$

## 5.7 | Security analysis

1. Authentication: According to Theorem 1 and Theorem 4, both type I or type II adversary cannot forge a valid signature and aggregate signature that satisfies the verification equation  $\phi_\lambda P = R_\lambda + h_\lambda(\delta_\lambda PK_\lambda + A_\lambda + \alpha_\lambda P_{pub})$  and  $\phi P = \sum_{\lambda=1}^n R_\lambda +$

$\sum_{\lambda=1}^n h_\lambda \delta_\lambda PK_\lambda + \sum_{\lambda=1}^n h_\lambda A_\lambda + \left( \sum_{\lambda=1}^n h_\lambda \alpha_\lambda \right) P_{pub}$ . Thus, the message is authenticity.

2. Anonymity: A vehicle can calculate its pseudonyms by using the hash chain set  $\tilde{t}_{y,\lambda} = \{\tilde{t}_{2,\lambda}, \tilde{t}_{3,\lambda}, \dots, \tilde{t}_{n,\lambda}\}$ . The pseudonym mechanism  $PID_{y,\lambda} = h_1(ID_\lambda || \tilde{t}_{y,\lambda} || \nabla || ts_{cur})$  is used in the proposed scheme. So vehicles cannot reveal the true identities of each other under pseudonyms.
3. Unlinkability: In this scheme, different random numbers  $k, r_\lambda, \beta_\lambda, \xi_\lambda$  and  $PID_{y,\lambda}$  are used to compute the private value  $SK_\lambda$ , signature of vehicle is not the same for two communication sessions. So any two signatures cannot be linked to the same vehicle by adversary.
4. Traceability: The real identity is concealed in its pseudonym  $PID_{y,\lambda}$ . The TM and RTSA can obtain the real identity  $ID_\lambda$  when needed. Because the RTSA has stored the tuple  $(\tilde{t}_{1,\lambda}, ID_\lambda)$  in its repository, it can compute all the valid pseudonyms by extracting the secret  $\tilde{t}_{1,\lambda}$ . However, it is impossible for an adversary to extract the secret value  $\tilde{t}_{1,\lambda}$  due to the one-way property of the hash function operation.
5. Resistance to attacks: In this scheme, system public key  $P_{pub}$  and public key  $PK_\lambda$  corresponding to the vehicle all are embedded in the verification process to resist the forgery attacks from type I and type II adversaries. A timestamp  $T_k$  is used by a vehicle  $\lambda$ . Even if the timestamp succeeds in the freshness check, it cannot satisfy the verification equation  $\phi_\lambda P = R_\lambda + h_\lambda(\delta_\lambda PK_\lambda + A_\lambda + \alpha_\lambda P_{pub})$  and  $\phi P = \sum_{\lambda=1}^n R_\lambda + \sum_{\lambda=1}^n h_\lambda \delta_\lambda PK_\lambda + \sum_{\lambda=1}^n h_\lambda A_\lambda + (\sum_{\lambda=1}^n h_\lambda \alpha_\lambda) P_{pub}$ . So the proposed scheme resists the replay attack.

## 6 | EFFICIENCY ANALYSIS

In this section, we will analyze the proposed scheme's security and efficiency. Table 2 shows the comparison results, including authentication, anonymity, unlinkability, without pairing, security for  $\mathcal{A}_1$ , and security for  $\mathcal{A}_2$ . Here, the entry "✓" indicates that the scheme meets the demand, and the entry "✗" indicates that the scheme does not satisfy the target. The above comparison shows that our scheme meets all security requirements, and the competitive scheme only meets some of the security features.

To clearly describe the comparison of computational efficiency, a simulation experiment was run on a computer equipped with an Intel Core i7-7700 CPU @ 3.6 GHz and a memory of 8 GB. We implemented this experiment with the pairing-based cryptography library in VC++ 6.0. For the pairing-based schemes, in order to provide a same security level as 1024-bit RSA, we use the supersingular curve  $y_2 = x_3 + x \bmod p$ , where the embedding degree is 2,  $q = 2^{159} + 2^{17} + 1$  represents a 160-bit Solinas prime, and  $p = 12qr - 1$  represents a 512-bit prime. With regard to the ECC-based scheme, so as to provide the equivalent security level, the Koblitz elliptic curve  $y_2 = x_3 + ax + b \bmod p$  defined on  $F_{2^{163}}$  was used to provide the ECC group.

**TABLE 2** The comparison of security properties in schemes

Scheme	Authentication	Anonymity	Unlinkability	Without pairing	Security against $\mathcal{A}_1$	Security against $\mathcal{A}_2$
10	✓	✓	✓	✓	✓	✗
12	✓	✓	✓	✗	✗	✓
15	✓	✓	✓	✓	✗	✗
32	✓	✓	✓	✗	✓	✓
11	✓	✗	✓	✓	✓	✓
13	✓	✗	✓	✗	✓	✓
Ours	✓	✓	✓	✓	✓	✓

$T_{bp}$	Pairing operation
$T_{bm}$	Scalar multiplication in bilinear pairing operation
$T_{ba}$	Addition in bilinear pairing operation
$T_{em}$	Scalar multiplication in ECC operation
$T_{ea}$	Addition in ECC operation
$T_h$	Hash operation
$ \mathbb{G} $	Size of a point in $\mathbb{G}$

**TABLE 3** Notations

Cryptographic operation	$T_{bp}$	$T_{bm}$	$T_{ba}$	$T_{em}$	$T_{ea}$	$T_h$
Time (ms)	11.9845	1.7090	0.0071	0.0321	0.0018	0.0085

**TABLE 4** Cryptographic operation and execution times

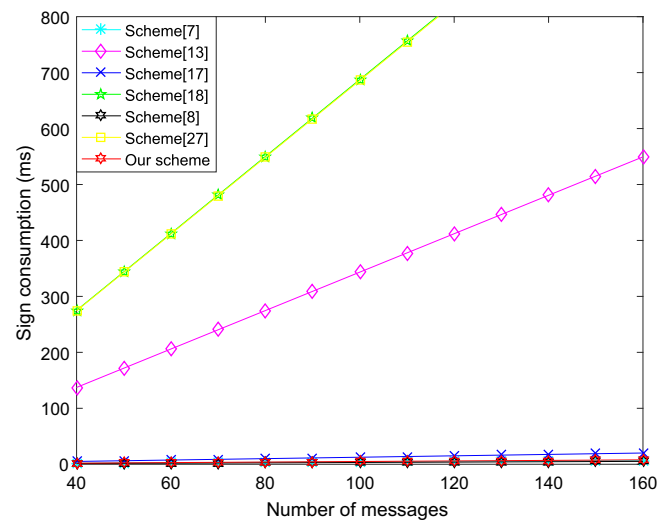
**TABLE 5** Comparison of our scheme with other related schemes

Scheme	Sign	Individual verify	Aggregate verify	SL	PKL
<sup>10</sup>	$T_{em} + T_{ea} + T_h$	$3T_{em} + 2T_{ea} + 2T_h$	$(n+2)T_{em} + (n+2)T_{ea} + 2nT_h$	$(n+1) G $	$n G $
<sup>12</sup>	$2T_{bm} + T_{ba} + T_h$	$3T_{bp} + T_{bm} + T_{ba} + 2T_h$	$3T_{bp} + nT_{bm} + nT_{ba} + 2nT_h$	$(n+1) G $	$n G $
<sup>15</sup>	$3T_{em} + 2T_{ea} + 3T_h$	$2T_{em} + T_{ea} + T_h$	$2T_{em} + T_{ea} + nT_h$	$ G $	$n G $
<sup>32</sup>	$4T_{bm} + 2T_{ba} + 3T_h$	$4T_{bp} + 3T_{bm} + 4T_h$	$4T_{bp} + 3nT_{bm} + 4nT_h$	$(n+1) G $	$n G $
<sup>11</sup>	$T_{em}$	$3T_{em} + 4T_{ea}$	$(3n+1)T_{em} + (4n-1)T_{ea}$	$(n+1) G $	$n G $
<sup>13</sup>	$4T_{bm} + 2T_{ba} + T_h$	$3T_{bp} + 3T_{bm} + T_{ba}$	$3T_{bp} + 2nT_{bm} + nT_{ba}$	$(n+1) G $	$n G $
Ours	$T_{em} + 2T_h$	$4T_{em} + 3T_{ea} + T_h$	$(n+2)T_{em} + (n+3)T_{ea} + nT_h$	$n G $	$n G $

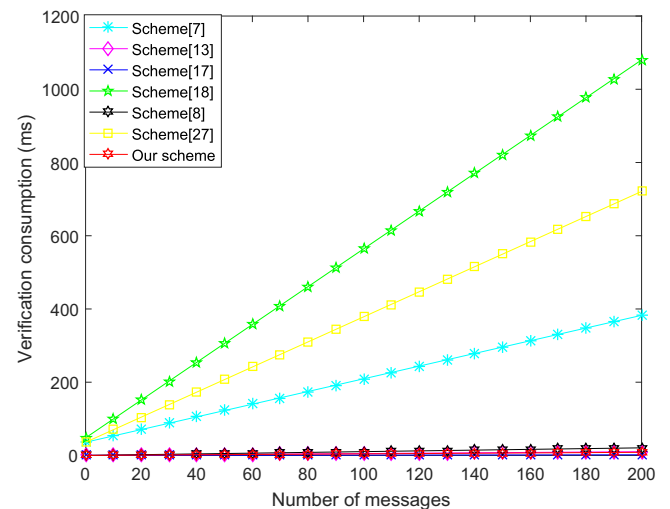
For convenience, Table 3 explains a few notations on execution time and their corresponding meanings. The running time of the operation is displayed in Table 4 by repeating the simulation experiment. Table 5 gives a theoretical assessment of signature length, signature cost, and verification cost. The comparison of the communication overhead between our scheme and the existing schemes is given in Figures 3 and 4.

From the results shown in Figures 3 and 4, we can see that compared with the previous studies,<sup>10-13,32</sup> the aggregate signature and aggregate verification cost of our scheme increase slowly. Specifically, with the increasing of the messages' number, the time consumption of the proposed scheme is almost constant and keeps below 20 ms. Although we can observe that our scheme has a higher communication time, it seems can be a reasonable expense for the significant improvement in security cost.

From Table 5, the size of our scheme's aggregate signature is much shorter than the scheme in aforementioned studies.<sup>10-13,32</sup> Hence, it is more suitable for our scheme to deploy on lightweight devices, which has limited computing



**FIGURE 3** Verification computation overhead of different schemes



**FIGURE 4** Sign computation overhead of different schemes

power and/or battery life. In addition, our scheme can resist the attacks from type I and type II adversaries, which is an advantage that does not exist in the existing works.

## 7 | CONCLUSION

In this paper, we discuss the security weaknesses of the scheme by Kamil and Ogundoyin.<sup>15</sup> After that, we introduced a new scheme with the capability to against the attacks from  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . The security properties of our scheme have been formally proven in the random oracle model. Comparing with existing works, the prominent performance advantages of our scheme have been demonstrated by extensive simulation results. Considering the computation cost, communication efficiency, and security properties, it is easy to see that our scheme is more suitable for the low-bandwidth environment.

## ORCID

Saru Kumari  <https://orcid.org/0000-0003-4929-5383>

## REFERENCES

1. Rappaport TS, Sun S, Mayzus R, et al. Millimeter wave mobile communications for 5G cellular: it will work! *IEEE Access*. 2013;1:335-349.
2. Andrews JG, Buzzi S, Choi W, et al. What will 5G be? *IEEE J Sel Areas Commun*. 2014;32(6):1065-1082.
3. Yu C, Lin B, Guo P, Zhang W, Li S, He R. Deployment and dimensioning of fog computing-based Internet of vehicle infrastructure for autonomous driving. *IEEE Internet Things J*. 2019;6:149-160.
4. Raya M, Hubaux JP. Securing vehicular ad hoc networks. *J Comput Secur*. 2007;15(1):39-68.
5. Boneh D, Gentry C, Lynn B, Shacham H. Aggregate and verifiably encrypted signatures from bilinear maps. In: *Advances in Cryptology - EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003. Proceedings*. Berlin, Germany: Springer; 2003:416-432.
6. Bhaskar R, Herranz J, Laguillaumie F. Aggregate designated verifier signatures and application to secure routing. *Int J Secur Netw*. 2007;2(3/4):192-201.
7. Lysyanskaya A, Micali S, Reyzin L, Shacham H. Sequential aggregate signatures from trapdoor permutations. In: *Advances in Cryptology hypen; EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings*. Berlin, Germany: Springer; 2004:74-90.
8. Wasef A, Shen X. ASIC: aggregate signatures and certificates verification scheme for vehicular networks. In: *Proceedings of the GLOBECOM 2009-2009 IEEE Global Telecommunications Conference; 2009; Honolulu, HI*.
9. Cheng X, Liu J, Wang X. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In: *Computational Science and Its Applications - ICCSA 2005: International Conference, Singapore, May 9-12, 2005. Proceedings, Part IV*. Berlin, Germany: Springer; 2005:1046-1054.
10. Cui J, Zhang J, Zhong H, Shi R, Xu Y. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*. 2018;451-452:1-15.
11. Du H, Wen Q, Zhang S. An efficient certificateless aggregate signature scheme without pairings for healthcare wireless sensor network. *IEEE Access*. 2019;7:42683-42693.
12. Horng S-J, Tzeng S-F, Huang P-H, Wang X, Li T, Khan MK. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Information Sciences*. 2015;317:48-66.
13. Wu L, Xu Z, He D, Wang X. New certificateless aggregate signature scheme for healthcare multimedia social network on cloud environment. *Secur Commun Netw*. 2018. <https://doi.org/10.1155/2018/2595273>
14. Xiong H, Guan Z, Chen Z, Li F. An efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*. 2013;219:225-235.
15. Kamil IA, Ogundoyin SO. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *J Inf Secur Appl*. 2019;44:184-200.
16. Al-Riyami SS, Paterson KG. Certificateless public key cryptography. In: *Advances in Cryptology - ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003. Proceedings*. Berlin, Germany: Springer; 2003:452-473. *Lecture Notes in Computer Science*; vol. 2894.
17. Huang X, Susilo W, Mu Y, Zhang F. On the security of certificateless signature schemes from Asiacypt 2003. In: *Cryptology and Network Security: 4th International Conference, CANS 2005, Xiamen, China, December 14-16, 2005. Proceedings*. Berlin, Germany: Springer; 2005:13-25.
18. Yum DH, Lee PJ. Generic construction of certificateless signature. In: *Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, July 13-15, 2004. Proceedings*. Berlin, Germany: Springer; 2004:200-211.
19. Hu BC, Wong DS, Zhang Z, Deng X. Key replacement attack against a generic construction of certificateless signature. In: *Information Security and Privacy: 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006. Proceedings*. Berlin, Germany: Springer; 2006:235-246.

20. Liu JK, Au MH, Susilo W. 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 (ASIACCS '07)*; 2007; Singapore.
21. Xiong H, Qin Z, Li F. An improved certificateless signature scheme secure in the standard model. *Fundamenta Informaticae*. 2008;88(1-2):193-206.
22. Xia Q, Xu CX, Yu Y. Key replacement attack on two certificateless signature schemes without random oracles. *Key Eng Mater*. 2010;439-440:1606-1611.
23. Choi KY, Park JH, Hwang JY, Lee DH. Efficient certificateless signature schemes. In: *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings*. Berlin, Germany: Springer; 2007:443-458.
24. Gorantla MC, Saxena A. An efficient certificateless signature scheme. In: *Computational Intelligence and Security: International Conference, CIS 2005, Xi'an, China, December 15-19, 2005. Proceedings, Part II*. Berlin, Germany: Springer; 2005:110-116.
25. Hu BC, Wong DS, Zhang Z, Deng X. Certificateless signature: a new security model and an improved generic construction. *Des Codes Cryptogr*. 2007;42:109-126.
26. Zhang Z, Wong DS, Xu J, Feng D. Certificateless public-key signature: security model and efficient construction. In: *Applied Cryptography and Network Security: 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006. Proceedings*. Berlin, Germany: Springer; 2006:293-308.
27. Gong Z, Long Y, Hong X, Chen K. Two certificateless aggregate signatures from bilinear maps. In: *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*; 2007; Qingdao, China.
28. Cheng L, Wen Q, Jin Z, Zhang H, Zhou L. Cryptanalysis and improvement of a certificateless aggregate signature scheme. *Information Sciences*. 2015;295:337-346.
29. He D, Tian M, Chen J. Insecurity of an efficient certificateless aggregate signature with constant pairing computations. *Information Sciences*. 2014;268:458-462.
30. Shen L, Ma J, Liu X, Wei F, Miao M. A secure and efficient ID-based aggregate signature scheme for wireless sensor networks. *IEEE Internet Things J*. 2016;4(2):546-554.
31. Gayathri NB, Thumbur G, Reddy PV, Muhammad ZUR. Efficient pairing-free certificateless authentication scheme with batch verification for vehicular ad-hoc networks. *IEEE Access*. 2018;6:31808-31819.
32. Kumar P, Kumari S, Sharma V, Li X, Sangaiah AK, Islam SH. Secure CLS and CL-AS schemes designed for VANETs. *J Supercomput*. 2018;75:3076-3098.
33. Qu Y, Mu Q. An efficient certificateless aggregate signature without pairing. *Int J Electron Secur Digit Forensics*. 2018;10(2):188-203.
34. Xiong H, Mei Q, Zhao Y. Efficient and provably secure certificateless parallel key-insulated signature without pairing for IIoT environments. *IEEE Syst J*. 2019;1-11. <https://doi.org/10.1109/JSYST.2018.2890126>

**How to cite this article:** Zhao Y, Hou Y, Wang L, Kumari S, Khan MK, Xiong H. An efficient certificateless aggregate signature scheme for the Internet of Vehicles. *Trans Emerging Tel Tech*. 2020;31:e3708. <https://doi.org/10.1002/ett.3708>