Received: 25 June 2019 | Revised: 19 October 2019 | Accepted: 27 October 2019

DOI: 10.1002/spy2.104

RESEARCH ARTICLE

WILEY

On the security of privacy-preserving authentication scheme with full aggregation in vehicular ad hoc network

Ismaila A. Kamil | **Sunday O. Ogundoyin**

Security, Privacy, and Communication (SPCOM) Research Group, Department of Electrical and Electronic Engineering, University of Ibadan, Ibadan, Nigeria

Correspondence
Sunday O. Ogundoyin, Security, Privacy, and Communication (SPCOM) Research Group, Department of Electrical and Electronic Engineering, University of Ibadan, Ibadan, Nigeria.
Email: honsybee@yahoo.com

Abstract

Certificateless aggregate signature (CLAS) scheme is a very important cryptographic technique used in many internet of things (IoT) applications like healthcare wireless sensor networks, industrial IoT, smart agriculture, and smart transportation to achieve privacy and integrity of transmitted information, and improved efficiency. Recently, a privacy-preserving authentication scheme based on CLAS scheme for secure communication in vehicular ad hoc network (VANET) which can achieve complete aggregation was proposed. The authors demonstrated that their scheme is semantically secure in the random oracle model based on the intractability of the computational Diffie-Hellman (CDH) problem under the consideration of type I and II attacks. However, by giving two concrete attacks, we show that the scheme is insecure in the standard security model. Consequently, we propose a fix by modifying the sign, verify, and aggregate-verify algorithms of the scheme. Afterwards, we demonstrate that with this modification, the improved scheme is semantically secure against forgery attacks in the random oracle model under the intractability of the CDH problem. An analysis of the performance of the proposed scheme and the related schemes shows the former is much more efficient and suitable for practical application.

KEYWORDS

aggregate signature, certificateless, computational Diffie-Hellman, privacy, random oracle, vehicular ad hoc networks

1 | INTRODUCTION

Intelligent transportation system (ITS) employs the applications of sensing, control, communication, and data analytic technologies to provide inventive services that can effectively address the traffic-related issues inherent in the traditional transportation system. In recent times, vehicles equipped with communication devices known as on-board units (OBUs) are emerging. Furthermore, roadside units (RSUs) are also being deployed along the roadside and at intersections to allow communication between vehicles and infrastructure. This new paradigm has innovated a self-organizing network known as vehicular ad hoc network (VANET). In general, as shown in Figure 1, a VANET consists of a trusted authority (TA), OBU-installed vehicles, and RSUs. Communication among vehicles, and between an RSU and an OBU is referred to as vehicle-to-vehicle (V2V) and is achieved using a dedicated short-range communication (DSRC) protocol[1]; while the TA, RSUs, and an application server (AS) communicate using a secure wired channel such as the Internet.[2] According

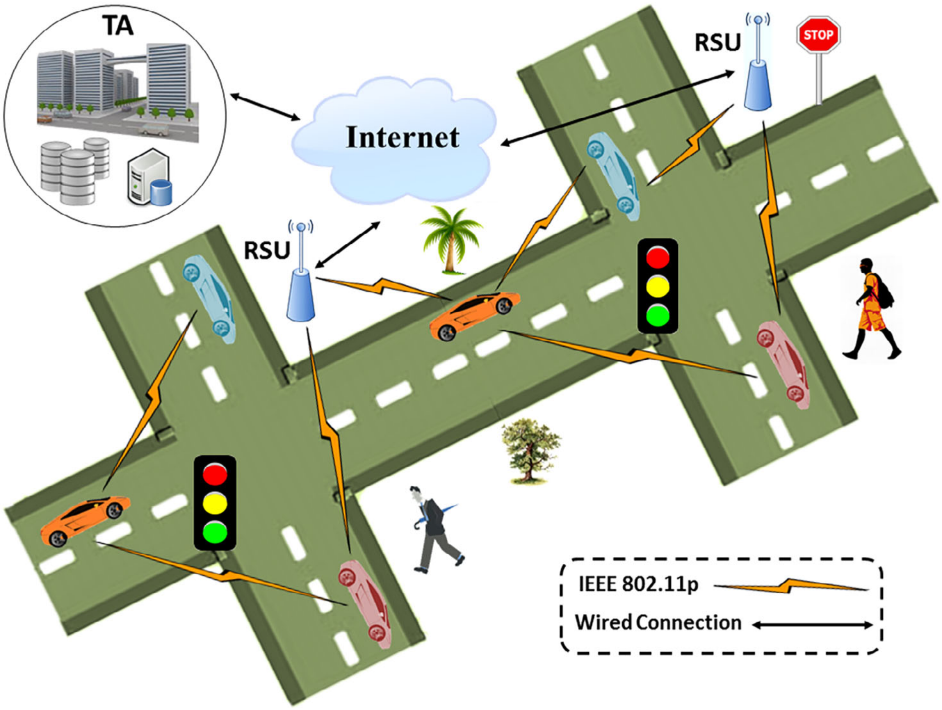

FIGURE 1 A simple architecture of vehicular ad hoc network

to VANET requirements, by using the DSRC protocol, each OBU frequently broadcasts traffic-related information every 100-300 ms.[2-7] RSUs can collect traffic-related data from OBUs, verify their authenticity and integrity, and then transmit the valid data to the AS for analysis.

Security and privacy are two key issues that need serious attention before the practical deployment of VANETs.[2-4,7-10] Secure vehicular communications can provide several benefits such as traffic monitoring, avoidance of traffic accidents, traffic decongestion, greenhouse emission reduction, and provision of traffic-related safety information such as coordinated collision warning, post-crash notifications, road hazard control notifications, etc.[4] Hence, communication in VANET must be protected against adversaries and unauthorized users. Moreover, privacy of users must be preserved since private information of users, like traveling routes, identity, location, and so forth, can expose them to unauthorized tracing and compromise.

Due to the nature of information transmitted in VANET, message authentication (achieved through digital signature) is a crucial security requirement in VANET to determine and filter unauthorized and misbehaving users. Many authentication techniques have been designed for different applications based on traditional public-key cryptography (PKC),[11-14] identity signature,[7,15] and group signature.[16] In PKC-based authentication techniques, a certificate authority needs to keep a huge amount of public-key certificates, making management of users uneasy. Thus, authentication schemes based on PKC have certificate management problem and high transmission overhead.[4] The group signature-based authentication technique can achieve strong anonymity,[17] but have high computation and communication costs,[4] while ID-based authentication schemes are inappropriate for private networks[17] due to key escrow problem.[4]

Certificateless PKC introduced in Reference 18 can address the key escrow problem in ID-based authentication schemes because a third party known as private key generator (PKG) is introduced which is responsible for issuing partial secret keys to users, while the secret keys are independently generated by the users. Aggregate signature can achieve reduced computation cost and transmission overhead since it allows a verifier to verify only one signature on n different messages instead of verifying n signatures. Thus, certificateless aggregate signature (CLAS) can address the certificate management problem, solve key escrow problem, and provide improved system efficiency.

In Reference 19, the authors proposed a pairing-based CLAS scheme and demonstrated that it is secure under the intractability of the computational Diffie-Hellman (CDH) problem. A new CLAS scheme without bilinear pairing was proposed in Reference 20 to address the efficiency issue in the previous schemes. Unfortunately, Du et al[21] showed that the scheme in Reference 20 is insecure against forgery attack by a type I adversary. Yang et al[22] put forward a short CLAS scheme based on bilinear pairing which can withstand coalition attacks. An unrestricted CLAS scheme with full aggregation was proposed in Reference 23. The authors demonstrated the security of their scheme against a normal-type

I and super-type II adversaries in the random oracle model. Kumar et al[24] also proposed a CLAS scheme for healthcare wireless sensor network (HWSN) with constant pairing computation and proved its security in the random oracle model under the hardness assumption of the CDH problem. However, it was revealed in Reference 25 that the scheme cannot withstand a signature forgery attacks by type II adversaries. Another CLAS scheme for HWSN based on pairings was developed in Reference 26 which relied on the intractability of the CDH problem. To address the security and efficiency issues in the previous schemes in HWSN environment, Xie et al[27] proposed a new CLAS scheme known as iCLAS. The scheme was constructed using elliptic curve cryptography and hash function operation which makes it more efficient than the previous schemes. It was demonstrated that iCLAS can prevent attacks from type I and II adversaries, if the discrete logarithm problem (DLP) is intractable.

Several CLAS schemes have been developed for practical applications in VANET in recent times. The authors in Reference 28 proposed a CLAS scheme for VANET which supported conditional privacy preservation. They showed that their scheme is secure in the random oracle model under the CDH problem assumption. However, it was pointed out in Reference 29 that the scheme in Reference 29 cannot withstand a type II attack. Kumar et al[30] designed a secure CLAS scheme for VANET. The scheme was constructed based on bilinear pairing and its security was demonstrated against both type I and II adversaries under the intractability of the CDH problem. However, the scheme is vulnerable to collision attack since an adversary and an RSU can collude to forge a vehicle's signature. Liu et al[31] proposed a batch verification scheme based a new CLAS in internet of vehicles. A state information is employed to allow a node join or leave the system dynamically. To address the efficiency and security issues in the existing CLAS schemes in VANET environments, Cui et al[6] developed a new CLAS scheme without bilinear pairing. The authors claimed that their scheme is semantically secure against type I and II adversaries in the random oracle model under the hardness assumption of the DLP. Although, the scheme addressed the efficiency issue in the previous CLAS schemes, it cannot withstand a signature forgery attack by a type II adversary.[4] Consequently, Kamil and Ogundoyin[4] proposed an improved scheme that addressed the security issue in Cui et al's scheme and achieved a better performance than the existing CLAS schemes. A pairing-free CLAS scheme for internet-of-things deployment was developed in Reference 32.

To address the efficiency issues in the previous schemes, Zhong et al[5] developed an authentication scheme using CLAS scheme which can achieve full aggregation for secure vehicle-to-infrastructure communications. The scheme does not rely on the conventional tamper-proof device and achieves reduced communication overhead. Moreover, the authors demonstrated the security of the scheme against both type I and type II adversaries using the random oracle model with the assumption that the (CDH) problem is hard to solve. Unfortunately, we find that their scheme is not secure against a signature-forgery attack by a type II adversary. Hence, a type II adversary can forge any message in this scheme. Consequently, we provide an improvement to remedy this flaw.

In this article, we put forward an improved CLAS scheme with full aggregation for VANET. Specifically, the main contributions of this work are as follows:

- We point out the security flaws in Zhong et al's scheme.
- We propose an improved scheme to address the weaknesses in Zhong et al's scheme.
- For improved efficiency, the proposed scheme implements full aggregation of certificateless signatures.
- We demonstrate that the proposed scheme is semantically unforgeable by type I and II adversaries in the random oracle model under the assumption that the CDH problem is intractable.

The rest of this research article is structured as follows. In Section 2, we present the preliminaries while Section 3 describes a CLS scheme. Section 4 discusses the security models of CLS and CLAS schemes. In Section 5, we review and analyze the Zhong et al's scheme, while an improvement is proposed in Section 6. We demonstrate the security and the performance of the proposed scheme in Section 7. We conclude the work in Section 8.

2 | PRELIMINARIES

2.1 | Bilinear pairings

Given that G_1 and G_2 are additive and multiplicative groups, respectively, a map $e: G_1 \times G_1 \rightarrow G_2$ is said to be a bilinear map provided that the following properties are satisfied.

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in \mathbb{Z}_n^*$.
- Nondegeneracy: There exist $P, Q \in G_1$ such that $e(P, Q) \neq 1$.
- Computability: An efficient algorithm exists to compute $e(P, Q)$ for every $P, Q \in G_1$.
- Symmetry: For every $P, Q \in G_1$, $e(P, Q) = e(Q, P)$.

2.2 | Computational assumption

The construction of the scheme is based on an assumption that the computational Diffie-Hellman (CDH) problem is hard.

- CDH problem: Given a point P of an additive group G_1 with order q and two points xP and yP, where $x, y \in \mathbb{Z}_q{}^*$ are unknown, compute $xyP \in G_1$.
- CDH assumption (CDHA): We say the $(\hat{t}, \epsilon)$-CDHA holds provided that no polynomial algorithm can solve the CDH problem in a time of at most $\hat{t}$ with a probability of at least ϵ.

3 | CERTIFICATELESS AGGREGATE SIGNATURE SCHEME

Generally, a CLAS scheme comprises the following polynomial-time algorithms.

1. Setup: The KGC takes security parameter λ as input and returns x, $P_{pub} = x \cdot P$, and *param* as its master key, public key, and public parameters, respectively.
2. Partial-Secret-Key-Gen: The KGC inputs the public parameters, master secret key x, and the identity RID_i of user i. It outputs the partial secret key psk_i and sends it to the user.
3. User-Key-Gen: A user i executes this algorithm by taking its real identity RID_i, partial secret key psk_i, public parameters, and random number a as input. It outputs sk_i and pk_i as the secret and public keys of the user, respectively.
4. Sign: A user i inputs its secret sk_i, partial secret key psk_i, public parameters, its identity RID_i, and a message m_i. It generates a signature σ_i on m_i.
5. Aggregate-Sign: An aggregator executes this algorithm. It takes a set of n signatures $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ on n messages $\{m_1, m_2, \ldots, m_n\}$ as input. It then generates an aggregate signature σ on the messages $\{m_1, m_2, \ldots, m_n\}$.
6. Aggregate-Verify: A verifier takes the public parameters, a set of n public keys $\{pk_1, pk_2, \ldots, pk_n\}$ of users with identities $\{RID_1, RID_2, \ldots, RID_n\}$, and the aggregate signature σ on the messages $\{m_1, m_2, \ldots, m_n\}$ as input. The outputs of the algorithm indicate whether the verification is valid or not.

4 | SECURITY MODEL CLS AND CLAS SCHEMES

Two categories of adversaries are considered: type I ($\mathcal{A}_1$) and type II ($\mathcal{A}_2$). $\mathcal{A}_1$ can replace the public or secret key of a user, but cannot obtain the master secret key of KGC. $\mathcal{A}_2$ can access the master secret key of KGC, but cannot replace a user's public or secret key.[20]

The security of a CLS or CLAS scheme is substantiated using two games played between an adversary $\mathcal{A}_1$ or $\mathcal{A}_2$ and a challenger C, in which $\mathcal{A}_1$ or $\mathcal{A}_2$ can simulate the following oracles.

1. Create-User: After receiving a public key request query of a user with identity RID_i, C sends pk_i of the user.
2. Reveal-Partial-Secret-Key: Upon receiving a partial secret key request query of a user with identity RID_i, C sends psk_i of the user.
3. Reveal-Secret-Key: After receiving a secret key request query of a user with identity RID_i, C sends sk_i.
4. Replace-Key: When C receives a public key replacement request of a user with identity RID_i, it replaces the pk_i with $pk_i{}'$.
5. Sign: On receiving this query, C returns σ_i as a signature of the user with identity RID_i on a message m_i.

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

- $\mathcal{C}$ simulates the Setup algorithm and generates x and *param* as the master key and the public parameters, respectively. It sends *param* to $\mathcal{A}_1$ and keeps x in its database.
- $\mathcal{A}_1$ simulates the Create-User, Reveal-Secret-Key, Reveal-Partial-Secret-Key, and Sign oracles.
- $\mathcal{A}_1$ generates σ_i as a signature of a user RID_i on message m_i.

We say $\mathcal{A}_1$ succeeds in game I provided that the three conditions below are met.

1. RID_i has never been given to the Reveal-Partial-Secret-Key oracle to get psk_i.
2. σ_i is a legal signature of user RID_i having pk_i as its public key.
3. (RID_i, m_i) has not been sent to the Sign oracle.

Game II: $\mathcal{C}$ and $\mathcal{A}_2$ play this game as follows.

- $\mathcal{C}$ executes the Setup algorithm so as to output x and *param* as the master secret key and public parameters, respectively. It then sends x *param* to $\mathcal{A}_2$.
- $\mathcal{A}_2$ executes the Create-User, Replace-Key, Reveal-Secret-Key, and Sign oracles.
- $\mathcal{A}_2$ generates σ_i as a signature of a user RID_i on a message m_i.

We say $\mathcal{A}_2$ wins game II provided the below three conditions are met.

1. RID_i has never been given to the Reveal-Secret-Key or Replace-Key oracles to get sk_i.
2. σ_i is a legal signature of user RID_i with a public key pk_i.
3. (RID_i, m_i) has not been submitted to the Sign oracle.

Theorem 1. *A CLS scheme is provably secure, provided no polynomial-time adversary $\mathcal{A}_1$ and $\mathcal{A}_2$ can win games I and II, respectively with a nonnegligible probability.*[4]

Game III: $\mathcal{C}$ and $\mathcal{A}_1$ played this game as described below.

- $\mathcal{C}$ simulates the Setup algorithm and generates x and *param* as the master secret key and public parameters, respectively. It submits *param* to $\mathcal{A}_1$ and stores x in its repository.
- $\mathcal{A}_1$ executes the Create-User, Reveal-Secret-Key, Reveal-Partial-Secret-Key, and sign oracles.
- $\mathcal{A}_1$ generates σ as an aggregate signature for a set of n users $\{RID_1, RID_2, \ldots, RID_n\}$ and the corresponding public keys $\{pk_1, pk_2, \ldots, pk_n\}$ on n distinct messages $\{m_1, m_2, \ldots, m_n\}$.

$\mathcal{A}_1$ wins game III provided that the three conditions below are met.

1. At least an element in the set $\{RID_1, RID_2, \ldots, RID_n\}$ has not been given to the Reveal-Partial-Secret-Key oracle to get psk_i.
2. σ represents a legal aggregate-signature on a message set $\{m_1, m_2, \ldots, m_n\}$ of users having identities $\{RID_1, RID_2, \ldots, RID_n\}$ and public keys $\{pk_1, pk_2, \ldots, pk_n\}$.
3. (RID_i, m_i) has not been sent to the Sign oracle.

Game IV: $\mathcal{C}$ and $\mathcal{A}_2$ played this game as described below.

- $\mathcal{C}$ executes the Setup algorithm and generates x and *param* as the master key and public parameters, respectively. It then sends x and *param* to $\mathcal{A}_2$.
- $\mathcal{A}_2$ executes the Create-User, Replace-Key, Reveal-Secret-Key, and Sign oracles.
- $\mathcal{A}_2$ generates σ as an aggregate signature for a set of n users $\{RID_1, RID_2, \cdots, RID_n\}$ and the corresponding public keys $\{pk_1, pk_2, \cdots, pk_n\}$ on a set of n messages $\{m_1, m_2, \cdots, m_n\}$.

TABLE 1 Notations

Notation	Description
G_1	Cyclic additive group
G_2	Cyclic multiplicative group
P	Generator of G_1
e	Bilinear map $e: G_1 \times G_2 \rightarrow G_2$
V_i	ith vehicle
R_j	jth RSU
s	Private key of PKG
P_{pub}	Public key of PKG
α	Private key of TRA
T_{pub}	Public key of TRA
PKG	Private key generator
TRA	Trace authority
RID_i	Real identity of V_i
ID_{R_j}	Identity of R_j
v_{pk_i}	Public key of V_i
v_{sk_i}	Secret key of V_i
v_{psk_i}	Partial private key of V_i
PID_i	Pseudo-identity of V_i
V_{P_i}	Validity period of PID_i
λ	Security parameter
$H_i(.)_{i=0,1,2,3}$	One-way hash function
m_i	Message signed by V_i
σ_i	Signature of V_i on m_i
t_i	Current timestamp used by V_i on m_i
$\|\|$	Concatenation operation
σ	Aggregate signature
$\oplus$	Exclusive (XOR) operation

$\mathcal{A}_2$ wins game IV provided that the below conditions are met.

1. At least an element of the set $\{RID_1, RID_2, \dots, RID_n\}$ has never been submitted to the Reveal-Secret-Key or Replace-Key oracles to get sk_i.
2. σ is a legal aggregate-signature on message set $\{m_1, m_2, \cdots, m_n\}$ of users $\{RID_1, RID_2, \dots, RID_n\}$ with the public keys $\{pk_1, pk_2, \dots, pk_n\}$.
3. (RID_i, m_i) has not been sent to the Sign oracle.

Theorem 2. *A CLAS scheme is provably-secure provided no polynomial-time adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ can win the games III and IV, respectively with a nonnegligible probability.*[4,20]

5 | REVIEW AND CRYPTANALYSIS OF ZHONG ET AL'S SCHEME

5.1 | Review of Zhong et al's CLAS scheme

In this subsection, we present a brief review of Zhong et al's scheme.[5] For better understanding, we give some notations used in Table 1 and the flowchart in Figure 2. The authors first made a high-level description and then give a full description of their scheme.

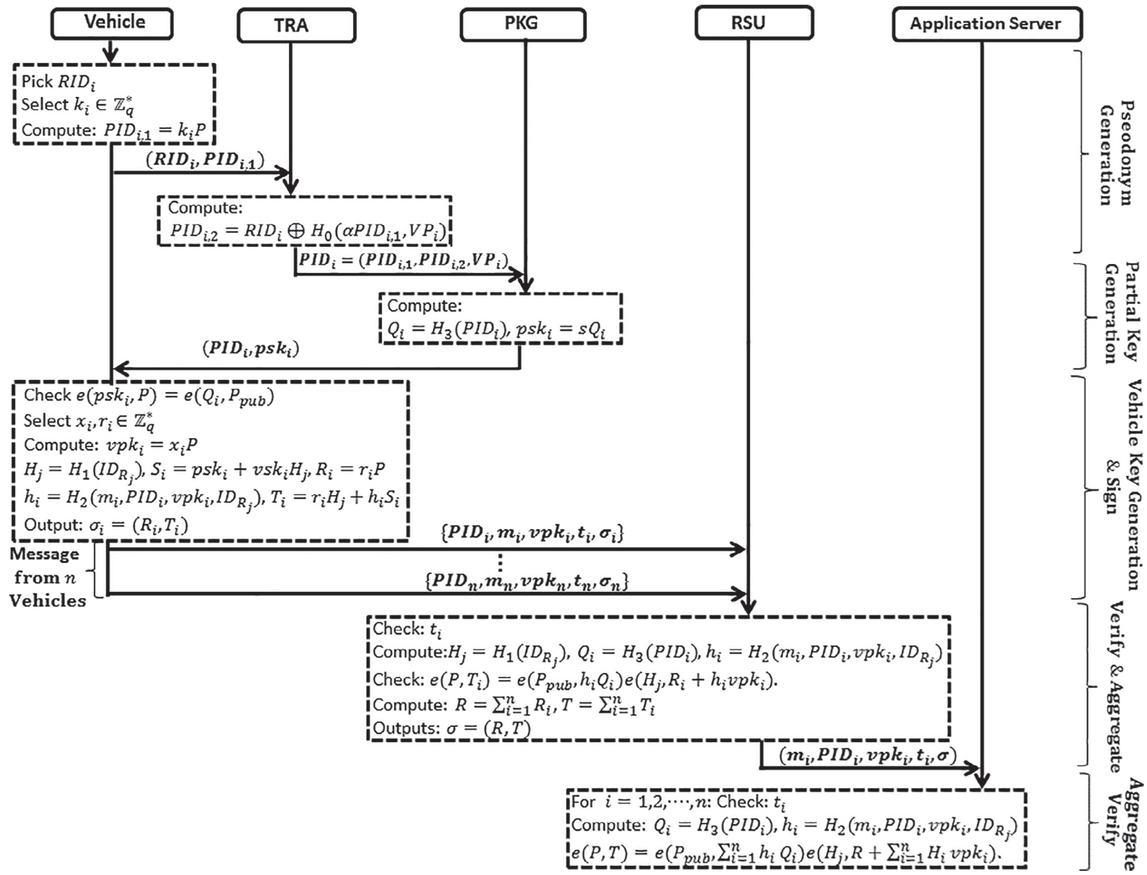

FIGURE 2 The flowchart of Zhong et al's scheme

5.1.1 | High-level description

The scheme is divided into eight algorithms as follows.

- System setup: The TAs take as input, a security parameter λ and generates the master-secret and system public keys, respectively. They then send the system public parameters to all the users.
- Pseudonym generation: After receiving the real identity RID_i of a vehicle i as input, the trace authority (TRA) returns the pseudonym to i.
- Partial-key generation: Upon receiving a vehicle's pseudonym PID_i as input, the KGC computes a partial-secret key psk_i and sends it the vehicle.
- Vehicle-key generation: Vehicle i picks a pseudo-identity PID_i and a number at random, and then returns the secret and public keys vsk_i and vpk_i, respectively.
- Sign: A vehicle utilizes the secret keys (psk_i, vsk_i) for signing a message $m_i \in \{0, 1\}^*$ and returns a certificateless signature σ_i.
- Verify: An RSU takes a signature-message pair $\sigma_i \parallel m_i$, a pseudonym PID_i, with the public key vpk_i as input, and outputs whether the signature is valid or otherwise.
- Aggregate: On receiving n signature-message pairs set, an RSU aggregates the signatures by computing $\sigma = \sum_{i=1}^{n} \sigma_i$.
- Aggregate verify: Upon receiving σ on n message-pseudonym-public key pairs, AS returns whether σ is valid or not.

5.1.2 | Scheme construction

The scheme is constructed using the following algorithms.

- System setup: The TAs input a security parameter λ and generate two groups G_1 and G_2, which are the cyclic additive and multiplicative groups, respectively, with the same prime order q and P is the generator of G_1. There exists a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The PKG selects $s \in \mathbb{Z}_q^*$ at random and calculates $P_{pub} = sP$. TRA selects a random $\alpha \in \mathbb{Z}_q^*$ and computes $T_{pub} = \alpha P$. TAs select four secure hash functions: $H_0 : \{0, 1\}^* \rightarrow G_1$, $H_1 : \{0, 1\}^* \rightarrow G_1$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and $H_3 : \{0, 1\}^* \rightarrow G_1$, and publish the public system parameters $\{q, G_1, G_2, e, P, P_{pub}, T_{pub}, H_0, H_1, H_2, H_3\}$.
- Pseudonym generation: A vehicle V_i picks a random $k_i \in \mathbb{Z}_q^*$, computes $PID_{i,1} = k_iP$, and returns $(RID_i, PID_{i,1})$ to TRA. On receiving $(RID_i, PID_{i,1})$, the TRA checks the validity of RID_i, computes $PID_{i,2} = RID_i \oplus H_0(\alpha PID_{i,1}, VP_i)$, and sends $PID_i = (PID_{i,1}, PID_{i,2}, VP_i)$ through a secure channel to PKG, where VP_i is the time-validity of PID_i.
- Partial key generation: The PKG computes $Q_i = H_3(PID_i)$ and $psk_i = sQ_i$, where psk_i is the partial secret key of V_i. It then sends (PID_i, psk_i) to V_i. V_i verifies the validity of $pski_i$ using the equation $e(psk_i, P) = e(Q_i, P_{pub})$.
- Vehicle key generation: V_i selects $x_i \in \mathbb{Z}_q^*$ at random, sets x_i as its private key vsk_i, and generates its public key as $vpk_i = x_iP$.
- Sign: The vehicle V_i computes $H_j = H_1(ID_{R_j})$, $S_i = psk_i + vsk_iH_j$, and stores it in the tamper-proof device (TPD). It picks $r_i \in \mathbb{Z}_q^*$, computes $R_i = r_iP$, $h_i = H_2(m_i, PID_i, vpk_i, ID_{R_j})$ and $T_i = r_iH_j + h_iS_i$, where m_i is the message to be signed. In the end, it outputs a signature $\sigma_i = (R_i, T_i)$ on $m_i \parallel t_i$, where t_i is the current timestamp. It then transmits $\{PID_i, m_i, vpk_i, t_i, \sigma_i\}$ to RSU.
- Verify: On receiving the packet $(PID_i, m_i, vpk_i, t_i, \sigma_i)$, an RSU R_j checks the freshness of the timestamp t_i, computes $H_j = H_1(ID_{R_j})$, and then stores H_j in its database. R_j then computes $Q_i = H_3(PID_i)$, $h_i = H_2(m_i, PID_i, vpk_i, ID_{R_j})$, and checks the validity of the equation $e(P, T_i) = e(P_{pub}, h_iQ_i)e(H_j, R_i + h_ivpk_i)$.
- Aggregate: Given a set of n vehicles $\{V_1, V_2, \ldots, V_n\}$ having pseudo-identities $\{PID_1, PID_2, \cdots, PID_n\}$, public keys $\{vpk_i, vpk_2, \ldots, vpk_n\}$ and the corresponding message-signature pairs $\{(m_1 \parallel t_1, \sigma_1 = (R_1, T_1)), (m_2 \parallel t_2, \sigma_2 = (R_2, S_2)), \ldots, (m_n \parallel t_n, \sigma_n = (R_n, T_n))\}$, R_j computes $R = \sum_{i=1}^{n} R_i$, $T = \sum_{i=1}^{n} T_i$, and outputs an aggregate signature $\sigma = (R, T)$.
- Aggregate verify: Upon receiving $\sigma = (R, T)$ and the corresponding pseudo-identities, public keys, and messages, the AS checks if the timestamp t_i, for $i = 1, 2, \cdots, n$ is fresh or not. If and only if the timestamp is valid, the application server computes $Q_i = H_3(PID_i)$, $h_i = H_2(m_i, PID_i, vpk_i, ID_{R_j})$, for $i = 1, 2, \ldots, n$, and checks if $e(P, T) = e\left(P_{pub}, \sum_{i=1}^{n} h_iQ_i\right) e\left(H_j, R + \sum_{i=1}^{n} H_ivpk_i\right)$.

5.2 | Cryptanalysis of Zhong et al's scheme

Zhong et al[5] demonstrated that their schemes are semantically secure against type II adversary. However, we demonstrate that their claim is untrue since there is a polynomial-time type II adversary $\mathcal{A}_2$ who can always succeed in games II and IV.

5.2.1 | Attack on Zhong et al's CLS scheme

We demonstrate an attack against the Zhong et al's CLS scheme by showing that a type II adversary $\mathcal{A}_2$ can always succeed in game II. Suppose (PID_i, m_i^*) is the target identity and message to be forged by $\mathcal{A}_2$. Hence, $\mathcal{A}_2$ cannot submit (PID_i, m_i^*) to the sign oracle. The attack is demonstrated below.

- Setup: A challenger C runs this algorithm. It generates the master secret key s and the public system parameters. It then returns s and parameters to $\mathcal{A}_2$.
- Queries: $\mathcal{A}_2$ picks a message m_i at random such that $m_i \neq m_i^*$, and then queries the sign oracle with (PID_i, m_i). Upon receiving this query, C outputs a valid signature $\sigma_i = (R_i, T_i)$, and

$$R_i = r_iP$$
$$T_i = r_iH_j + h_iS_i$$

where $H_j = H_i(ID_{R_j})$, $S_i = psk_i + vsk_iH_j$, $h_i = H_2(m_i, PID_i, vpk_i, ID_{R_j})$, $psk_i = sQ_i$, $Q_i = H_3(PID_i)$, where ID_{R_j} and vpk_i are the identity of RSU and public key of PID_i, respectively. So, $T_i = r_iH_j + h_i(psk_i + vsk_iH_j)$.

Since $\mathcal{A}_2$ has the master secret key s, it can calculate

$$\beta = T_i - h_ipsk_i = r_iH_j + h_ivsk_iH_j$$

Note that $\mathcal{A}_2$ is unaware of the secret key vsk_i and random value r_i. Then, it can compute

$$\beta^* = \frac{1}{h_i}\beta_i = \frac{r_i H_j}{h_i} + vsk_i H_j$$

where $\frac{1}{h_i}$ satisfies $\frac{1}{h_i} \cdot h_i \equiv 1 \ mod \ q$.

Forgery: $\mathcal{A}_2$ outputs a forged signature $\sigma_i{}^*$ on a message $m_i{}^*(m_i{}^* \neq m_i)$ with identity PID_i and the public key vpk_i as follows.

Computes

$$\begin{aligned} h_i{}^* &= H_2(\mathrm{m}_i{}^*, PID_i, vpk_i, ID_{R_j}) \\ T_i{}^* &= h_i{}^* psk_i + h_i{}^* \beta^* \\ &= h_i{}^* psk_i + \frac{h_i{}^*}{h_i} r_i H_j + h_i{}^* vsk_i H_j \end{aligned}$$

Sets $R_i{}^* = \frac{h_i{}^*}{h_i} R_i$ and outputs $\sigma_i{}^* = (R_i{}^*, T_i{}^*)$ as a forged signature on message $m_i{}^*$

- Verify: The verifier (R_j) computes $h_i{}^* = H_2(m_i{}^*, PID_i, vpk_i, ID_{R_j})$, $H_j = H_i(ID_{R_j})$, $Q_i = H_3(PID_i)$, and checks the validity of the equation

$$\begin{aligned} e(P, T_i{}^*) &= e\left(P, h_i{}^* psk_i + \tfrac{h_i{}^*}{h_i} r_i H_j + h_i{}^* vsk_i H_j\right) \\ &= e(P, h_i{}^* psk_i) e\left(P, \tfrac{h_i{}^*}{h_i} r_i H_j\right) e(P, h_i{}^* vsk_i H_j) \\ &= e(P, h_i{}^* s\mathrm{Q}_i) e\left(\tfrac{h_i{}^*}{h_i} r_i P, H_j\right) e(vsk_i P, h_i{}^* H_j) \\ &= e(sP, h_i{}^* Q_i) e\left(\tfrac{h_i{}^*}{h_i} R_i, H_j\right) e(vpk_i, h_i{}^* H_j) \\ &= e(P_{pub}, h_i{}^* Q_i) e(R_i{}^*, H_j) e(H_j, h_i{}^* vpk_i) \\ &= e(P_{pub}, h_i{}^* Q_i) e(H_j, R_i{}^* + h_i{}^* vpk_i) \end{aligned}$$

Obviously, $\sigma_i = (R_i{}^*, T_i{}^*)$ is a valid signature on the message $m_i{}^*$. Note that $\mathcal{A}_2$ does not have the knowledge of r_i and vsk_i, and has never submitted $(PID_i, m_i{}^*)$ to the sign oracle. Hence, $\mathcal{A}_2$ wins game II. Hence, Zhong et al's CLS scheme is not semantically secure against the type II adversary.

5.2.2 | Attack on Zhong et al's CLAS scheme

We demonstrate an attack against the Zhong et al's CLAS scheme by showing that a type II adversary $\mathcal{A}_2$ can always succeed in game IV. Suppose there exist n vehicles having identities $\{PID_1, PID_2, \ldots, PID_n\}$ and the corresponding public keys $\{vpk_1, vpk_2, \ldots, vpk_n\}$. The attack is described as follows.

1. Setup: A challenger C runs the setup algorithm, generates the master secret key s and the public parameters, and then sends them to $\mathcal{A}_2$.
2. Queries: Because $\mathcal{A}_2$ possesses s, it can simulate the attack described in Section 5.2.2. In other words, for each vehicle in the network, $\mathcal{A}_2$ can obtain n forged message-signature pairs $\{(m_1{}^*, \sigma_1{}^* = (R_1{}^*, T_1{}^*)), (m_2{}^*, \sigma_2{}^* = (R_2{}^*, T_2{}^*)), \ldots, (m_n{}^*, \sigma_n{}^* = (R_n{}^*, T_n{}^*))\}$.
3. Forgery: $\mathcal{A}_2$ outputs $\sigma^* = (R^*, T^*)$ as a forged CLAS, where $R^* = \sum_{i=1}^n R_i{}^*$ and $T^* = \sum_{i=1}^n T_i{}^*$, for $1 \le i \le n$.
4. Aggregate Verify: On receiving the forged CLAS $\sigma^* = (R^*, T^*)$, the verifier (application server) computes $h_i{}^* = H_2(m_i{}^*, PID_i, vpk_i, ID_{R_j})$, $H_j = H_i(ID_{R_j})$, $Q_i = H_3(PID_i)$, for $1 \le i \le n$, and checks the validity of the following equation.

$$\begin{aligned} e(P, T^*) &= e\left(P, \sum_{i=1}^n h_i{}^* psk_i + \sum_{i=1}^n \frac{h_i{}^*}{h_i} r_i H_j + \sum_{i=1}^n h_i{}^* vsk_i H_j\right) \\ &= e\left(P, \sum_{i=1}^n h_i{}^* psk_i\right) e\left(P, \sum_{i=1}^n \frac{h_i{}^*}{h_i} r_i H_j\right) e\left(P, \sum_{i=1}^n h_i{}^* vsk_i H_j\right) \end{aligned}$$

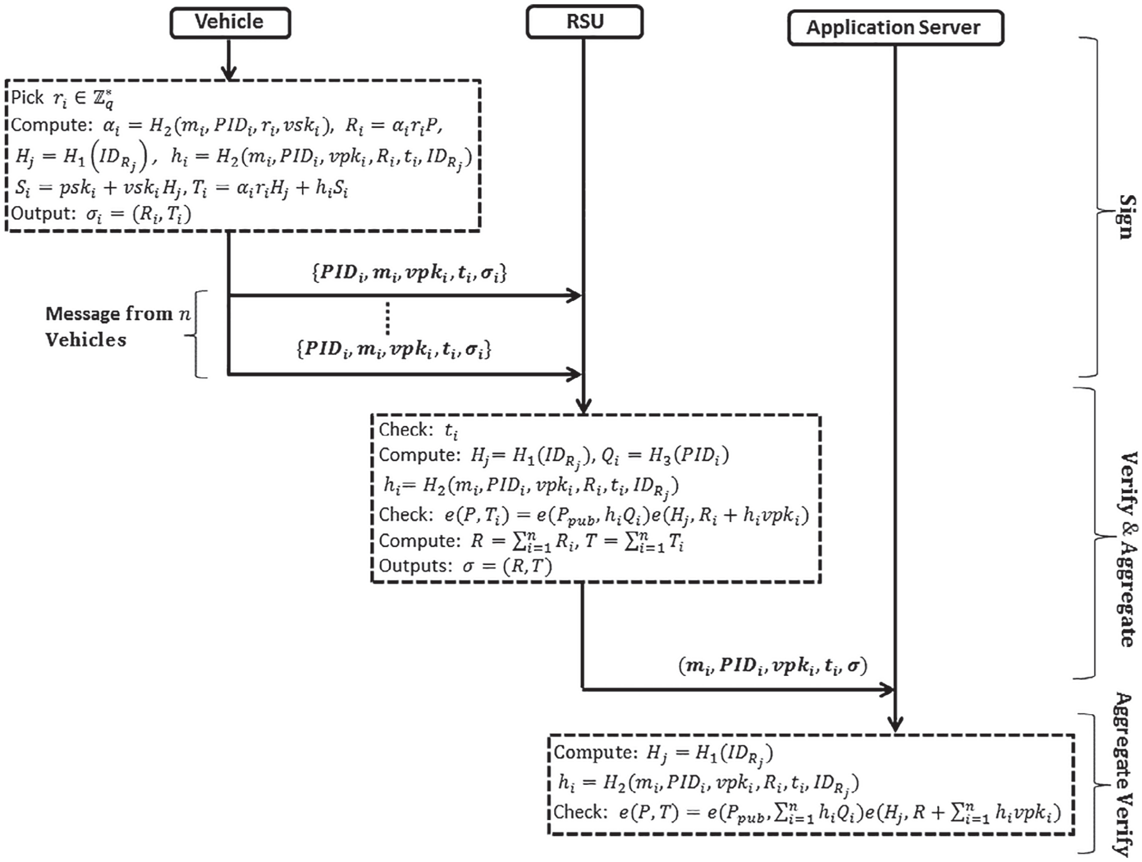

FIGURE 3 The flowchart of the improved scheme

$$
\begin{aligned}
&= e\left(P, \sum_{i=1}^{n} h_i^* sQ_i\right) e\left(\sum_{i=1}^{n} \frac{h_i^*}{h_i} r_i P, H_j\right) e\left(\sum_{i=1}^{n} vsk_i h_i^* P, H_j\right) \\
&= e\left(sP, \sum_{i=1}^{n} h_i^* Q_i\right) e\left(\sum_{i=1}^{n} \frac{h_i^*}{h_i} R_i, H_j\right) e\left(\sum_{i=1}^{n} vpk_i h_i^*, H_j\right) \\
&= e\left(P_{pub}, \sum_{i=1}^{n} h_i^* Q_i\right) e\left(H_j, \sum_{i=1}^{n} R_i^*\right) e\left(H_j, \sum_{i=1}^{n} vpk_i h_i^*\right) \\
&= e\left(P_{pub}, \sum_{i=1}^{n} h_i^* Q_i\right) e\left(H_j, \sum_{i=1}^{n} R_i^* + \sum_{i=1}^{n} vpk_i h_i^*\right) \\
&= e\left(P_{pub}, \sum_{i=1}^{n} h_i^* Q_i\right) e\left(H_j, R^* + \sum_{i=1}^{n} vpk_i h_i^*\right)
\end{aligned}
$$

Hence, $\sigma^* = (R^*, T^*)$ is a valid CLAS on the set of messages $\{m_1^*, m_2^*, \ldots, m_n^*\}$ of vehicles with identities $\{PID_1, PID_2, \ldots, PID_n\}$ and the corresponding public keys $\{vpk_1, vpk_2, \ldots, vpk_n\}$. It can be observed that, for any $i \in \{1, 2, \ldots, n\}$, $\mathcal{A}_2$ does not know r_i and vsk_i, and has never submitted (PID_i, m_i^*) to the sign oracle. Thus, $\mathcal{A}_2$ wins game IV. Therefore, Zhong et al's CLAS scheme is not secure against the type II adversary.

6 | IMPROVEMENT ON ZHONG ET AL'S SCHEME

From our analysis, $\mathcal{A}_2$ could utilize $R_i = r_i P$, $T_i = r_i H_j + h_i S_i$, and h_i to calculate $\beta^* = \frac{r_i}{h_i} H_j + vsk_i H_j$. To resist this attack, we just need to stop $\mathcal{A}_2$ from obtaining $\beta^* = \frac{r_i}{h_i} H_j + vsk_i H_j$. So, we modify the sign, verify, and aggregate verify algorithms of the scheme in Reference 5 as follows. The flowchart of the improved scheme is shown in Figure 3.

Sign: Signing a message m_i requires the execution of the following algorithms by a vehicle V_i with secret key vsk_i, partial secret key psk_i, and pseudo-identity PID_i.

- Computes $H_j = H_1(ID_{R_j})$ and $S_i = psk_i + vsk_iH_j$,. Note that H_j and S_i are calculated once when the vehicle enters the RSU's coverage.
- Picks $r_i \in \mathbb{Z}_q^*$ at random and calculates $\alpha_i = H_2(m_i, PID_i, r_i, vsk_i)$ and $R_i = \alpha_i r_i P$.
- Computes $h_i = H_2(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j})$ and $T_i = \alpha_i r_i H_j + h_i S_i$
- Outputs a signature $\sigma_i = (R_i, T_i)$ on $m_i \parallel t_i$, where t_i is the current timestamp.
- Sends $\{PID_i, m_i, vpk_i, t_i, \sigma_i\}$ to the corresponding RSU.

Verify: After receiving $\{PID_i, m_i, vpk_i, t_i, \sigma_i\}$, the verifier (RSU) checks the freshness of t_i and then performs the following.

- Computes $H_j = H_1(ID_{R_j})$, $Q_i = H_3(PID_i)$, and $h_i = H_2(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j})$. Note that, H_j is only computed once by R_j.
- Verifies the validity of the equation $e(P, T_i) = e(P_{pub}, h_iQ_i)e(H_j, R_i + h_ivpk_i)$.

The correctness is as follows:

$$\begin{aligned}
e(P, T_i) &= e(P, \alpha_i r_i H_j + h_i psk_i + h_i vsk_i H_j) \\
&= e(P, \alpha_i r_i H_j)e(P, h_i psk_i)e(P, h_i vsk_i H_j) \\
&= e(P, h_i s Q_i)e(P, \alpha_i r_i H_j)e(P, h_i vsk_i H_j) \\
&= e(sP, h_i Q_i)e(\alpha_i r_i P, H_j)e(vsk_i P, h_i H_j) \\
&= e(P_{pub}, h_i Q_i)e(R_i, H_j)e(vpk_i, h_i H_j) \\
&= e(P_{pub}, h_i Q_i)e(H_j, R_i)e(H_j, h_i vpk_i) \\
&= e(P_{pub}, h_i Q_i)e(H_j, R_i + h_i vpk_i)
\end{aligned}$$

Aggregate: On receiving a set of n message-signature pairs $\{(m_1 \parallel t_1, \sigma_1 = (R_1, T_1)), (m_2 \parallel t_2, \sigma_2 = (R_2, S_2)), \dots, (m_n \parallel t_n, \sigma_n = (R_n, T_n))\}$ from n vehicles having pseudo-identities $\{PID_1, PID_2, \dots, PID_n\}$, and the corresponding public keys $\{vpk_i, vpk_2, \dots, vpk_n\}$, R_j computes $R = \sum_{i=1}^n R_i$, $T = \sum_{i=1}^n T_i$ and outputs a CLAS $\sigma = (R, T)$.

Aggregate verify: To verify a CLAS $\sigma = (R, T)$, where $R = \sum_{i=1}^n R_i$, $T = \sum_{i=1}^n T_i$, for $1 \le i \le n$, the verifier (application server) does the following.

- Computes $H_j = H_1(ID_{R_j})$, $h_i = H_2(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j})$.
- Verifies the equation $e(P, T) = e\left(P_{pub}, \sum_{i=1}^n h_iQ_i\right) e\left(H_j, R + \sum_{i=1}^n h_ivpk_i\right)$, and accepts the CLAS if and only if the equation holds.

The correctness is as follows:

$$\begin{aligned}
e(P, T) &= e\left(P, \sum_{i=1}^n T_i\right) \\
e(P, T) &= e\left(P, \sum_{i=1}^n (\alpha_i r_i H_j + h_i psk_i + h_i vsk_i H_j)\right) \\
&= e\left(P, \sum_{i=1}^n \alpha_i r_i H_j\right) e\left(P, \sum_{i=1}^n h_i psk_i\right) e\left(P, \sum_{i=1}^n h_i vsk_i H_j\right) \\
&= e\left(P, \sum_{i=1}^n h_i s Q_i\right) e\left(P, \sum_{i=1}^n \alpha_i r_i H_j\right) e\left(P, \sum_{i=1}^n h_i vsk_i H_j\right) \\
&= e\left(sP, \sum_{i=1}^n h_i Q_i\right) e\left(\sum_{i=1}^n \alpha_i r_i P, H_j\right) e\left(H_j, \sum_{i=1}^n h_i vsk_i P\right) \\
&= e\left(P_{pub}, \sum_{i=1}^n h_i Q_i\right) e\left(\sum_{i=1}^n R_i, H_j\right) e\left(H_j, \sum_{i=1}^n h_i vpk_i\right) \\
&= e\left(P_{pub}, \sum_{i=1}^n h_i Q_i\right) \mathrm{e}(R, H_j) e\left(H_j, \sum_{i=1}^n h_i vpk_i\right) \\
&= e\left(P_{pub}, \sum_{i=1}^n h_i Q_i\right) e\left(H_j, R + \sum_{i=1}^n h_i vpk_i\right)
\end{aligned}$$

Based on a valid signature $\sigma_i = (R_i, T_i)$ and the partial secret key psk_i, the adversary $\mathcal{A}_2$ could calculate $\beta = T_i - h_i psk_i = \alpha_i r_i H_j + h_i vsk_i H_j$. However, it cannot succeed in the forgery since β is protected by h_i. This means that $\mathcal{A}_2$ cannot generate T_i^* for another message m_i^*. So, the improved scheme could resist the type II attack discussed in section IV.

7 | SECURITY ANALYSIS AND PERFORMANCE EVALUATION

7.1 | Security proof

We prove that the proposed scheme is semantically unforgeable in the random oracle model based on the intractability assumption of the CDH problem. We utilize two games played between a challenger C and two time-polynomial adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ to demonstrate that the proposed scheme is provably secure.

Theorem 3. *The proposed scheme is existentially unforgeable against an adaptively chosen-message attack by a type I adversary in the random oracle model under the intractability of the CDH problem.*

Proof. Given a random instance $(P, X = aP, Y = bP)$ of the CDH problem, where $a, b \in \mathbb{Z}_q^*$ are unknown to $\mathcal{A}_1$. Suppose $\mathcal{A}_1$ has an advantage ϵ in forging a valid signature in the proposed scheme in time t_c, then there exists an algorithm which acts as a challenger C that can compute $abP \in G_1$. If $\mathcal{A}_1$ could forge a valid signature in our scheme by making $\{q_{H_i}\}_{i=1,2,3}$ queries to H_i-Queries, q_{PSK} to Partial-Secret-Key-Queries, q_{SK} to Secret-Key-Queries, q_{PK} to Public-Key-Queries, and q_{sig} to Sign-Queries; then, C can solve the CDH problem with a probability of at least $\frac{\epsilon}{q_{PSK}e}$ within a time $t^{\ddagger} = t_c + t_m(q_{H_i} + q_{PSK} + q_{SK} + q_{PK} + q_{sig})$, where e is the base of the natural logarithm and t_m is the time taken to perform a scalar multiplication operation in G_1. $\mathcal{A}_1$ interacts with C as follows:

Setup: C picks a challenged identity PID_i^*, sets $P_{pub} = X$, and submits the public parameters $params = \{q, G_1, G_2, e, P, P_{pub}, T_{pub}, H_0, H_1, H_2, H_3\}$ to $\mathcal{A}_1$. It also maintains four lists L_{H_1}, L_{H_2}, L_{H_3} and L_K. It should be noted that each of the hash function H_1, H_2, and H_3 is considered a random oracle and $\mathcal{A}_1$ can perform any oracle query in the game process.

C manages the following lists which are initially empty.

L_{H_1} contains the tuple $(ID_{R_j}, \alpha_j, H_j)$.

L_{H_2} contains the tuple $\{(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j}, h_i), (m_i, PID_i, r_i, vsk_i, \alpha_i)\}$

L_{H_3} contains the tuple (PID_i, μ_i, Q_i)

L_K contains the tuple $(PID_i, psk_i, vsk_i, vpk_i)$

H_1-**Queries**: When $\mathcal{A}_1$ makes a query on ID_{R_j}, C checks if the list L_{H_1} contains the tuple $(ID_{R_j}, \alpha_j, H_j)$. If so, it returns H_j to $\mathcal{A}_1$; otherwise, it picks $\alpha_i \in \mathbb{Z}_q^*$ at random, sets $H_j = \alpha_j X$, inserts $(ID_{R_j}, \alpha_j, H_j)$ into L_{H_1} and returns H_j to $\mathcal{A}_1$.

H_2-**Queries**: When $\mathcal{A}_1$ makes a query on H_2-oracle, C checks if the list L_{H_2} contains the tuple $\{(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j}, h_i), (m_i, PID_i, r_i, vsk_i, \alpha_i)\}$. If so, it returns h_i or α_i to $\mathcal{A}_1$; otherwise, it picks at random $h_i, \alpha_i \in \mathbb{Z}_q^*$, inserts it in L_{H_2}, and sends h_i or α_i to $\mathcal{A}_1$.

H_3-**Queries**: When $\mathcal{A}_1$ submits this query on PID_i, C checks if the list L_{H_3} contains the tuple (PID_i, μ_i, Q_i). If so, it returns Q_i to $\mathcal{A}_1$; otherwise, if $PID_i = PID_i^*$, C picks at random $\mu_i \in \mathbb{Z}_q^*$, computes $Q_i = \mu_i Y \in G_1$, inserts it in L_{H_3}, and returns it to $\mathcal{A}_1$. If $PID_i \neq PID_i^*$, C picks at random $\mu_i \in \mathbb{Z}_q^*$, computes $Q_i = \mu_i P \in G_1$, inserts it in L_{H_3} and returns Q_i to $\mathcal{A}_1$.

Partial-Secret-Key-Queries: When $\mathcal{A}_1$ submits this query with an identity PID_i, if $PID_i = PID_i^*$, C aborts the game process. If $PID_i \neq PID_i^*$ and list L_K contains $(PID_i, psk_i, vsk_i, vpk_i)$, then C confirms if $psk_i = \perp$. If $psk_i \neq \perp$, C returns psk_i to $\mathcal{A}_1$. If $psk_i = \perp$, C checks L_{H_3} and returns psk_i to $\mathcal{A}_1$. If the list L_K does not include the tuple $(PID_i, psk_i, vsk_i, vpk_i)$, C sets $psk_i = \perp$, checks the list L_{H_3}, sets $psk_i = \mu_i P_{pub} = \mu_i X \in G_1$, and returns psk_i to $\mathcal{A}_1$.

Secret-Key-Queries: When $\mathcal{A}_1$ submits this query, C checks if the list L_K contains the tuple $(PID_i, psk_i, vsk_i, vpk_i)$. If so, it checks if $vsk_i = \perp$. If $vsk_i \neq \perp$, C returns vsk_i to $\mathcal{A}_1$. If $vsk_i = \perp$, C picks at random $d_i \in \mathbb{Z}_q^*$, sets $vpk_i = d_i P$, and submits d_i to $\mathcal{A}_1$. If the tuple $(PID_i, psk_i, vsk_i, vpk_i)$ is not found in L_K, C sets $vsk_{=}\perp$. If $vpk_i = \perp$, C picks a random $d_i \in \mathbb{Z}_q^*$, sets $vpk_i = d_i P$, submits vsk_i to $\mathcal{A}_1$, and inserts the tuple $(PID_i, psk_i, vsk_i, vpk_i)$ into L_K.

Public-Key-Queries: When $\mathcal{A}_1$ makes this query on (PID_i, vpk_i), C checks if the list L_K contains $(PID_i, psk_i, vsk_i, vpk_i)$. If so, it sets $vsk_i = \perp$ and $vpk_i = vpk_i^*$, and updates the list L_K. Otherwise, it sets $vsk_i = \perp$, $vpk_i = vpk_i^*$, $psk_i = \perp$, and updates the list L_K.

Sign-Queries: When $\mathcal{A}_1$ makes a query on (PID_i, m_i), C checks the lists L_{H_1}, L_{H_2}, L_{H_3} and L_K, and does as follows.

- If the tuple $(PID_i, psk_i, vsk_i, vpk_i)$ is present in L_K, it checks vsk_i. If $vsk_i \neq \perp$, it returns vpk_i to $\mathcal{A}_1$. Otherwise, it executes the Public-Key-Queries to obtain $vpk_i = d_iP$, where $d_i \in \mathbb{Z}_q^*$.
- If the tuple $(PID_i, psk_i, vsk_i, vpk_i)$ is not present in L_K, C performs the Public-Key-Queries to generate (vsk_i, vpk_i) pair, inserts the tuple $(PID_i, psk_i, vsk_i, vpk_i)$ in L_K.

Forgery: To generate $\sigma_i = (R_i, T_i)$, C executes the following.

- If $PID_i = PID_i^*$, C picks at random $d_i \in \mathbb{Z}_q^*$, sets $R_i = h_iY$, $T_i = h_i\mu_iX + \alpha_jh_id_iX + \alpha_jh_ibX$, and returns $\sigma_i = (R_i, T_i)$ to $\mathcal{A}_1$. $\sigma_i = (R_i, T_i)$ can be verified as follows.

$$\begin{aligned} e(P,T) &= e(P, h_i\mu_iX + \alpha_jh_id_iX + \alpha_jh_ibX) \\ &= e(P, h_i\mu_iX)e(P, \alpha_jh_id_iX)e(P, \alpha_jh_ibX) \\ &= e(X, h_i\mu_iP)e(\alpha_jX, h_id_iP)e(\alpha_jX, h_ibP) \\ &= e(X, h_iQ_i)e(H_j, h_ivpk_i)e(H_j, h_iY) \\ &= e(X, h_iQ_i)e(H_j, h_ivpk_i)e(H_j, R_i) \\ &= e(P_{pub}, h_iQ_i)\mathrm{e}(H_j, R_i + h_ivpk_i) \end{aligned}$$

- If $PID_i \neq PID_i^*$, C picks at random $r_i, d_i \in \mathbb{Z}_q^*$, sets $R_i = r_iP$, $T_i = h_ipsk_i + \alpha_jX(h_id_i + r_i)$, and returns $\sigma_i = (R_i, T_i)$ to $\mathcal{A}_1$.

According to Forking lemma,[33] C can produce another valid signature $\sigma_i^* = (R_i, T_i^*)$ with the same random tape but different choice of H_2-oracle. Hence, we can have these two equations.

$$T_i = r_iH_j + h_i(psk_i + d_iH_j) \tag{1}$$

$$T_i = r_iH_j + h_i^*(psk_i + d_iH_j) \tag{2}$$

From Equations (1) and (2), and setting $H_j = K$ and $vsk_i = d_i$, we have

$$\begin{aligned} T_i - T_i^* &= h_i(psk_i + d_iK) - h_i^*(psk_i + d_iK) \\ &= h_ipsk_i + h_id_iK - h_i^*psk_i - h_i^*d_iK \\ &= (h_i - h_i^*)psk_i + (h_i - h_i^*)d_iK \end{aligned}$$

We set $psk_i = aQ_i$, $Q_i = \mu_iY$, and $Y = bP$. Thus, we have

$$T_i - T_i^* = (h_i - h_i^*)\mu_iabP + (h_i - h_i^*)d_iK \tag{3}$$

Therefore, C returns $abP = \{(T_i - T_i^*) - (h_i - h_i^*)d_iK\}(h_i - h_i^*)^{-1}\mu_i^{-1}$ as the solution to the CDH problem with a probability of at least $\frac{\epsilon}{q_{cu}e}$ analyzed as follows.

There are three events required for C to succeed in the game:

E_1: C does not abort the Partial-Secret-Key-Queries.

E_2: $\mathcal{A}_1$ forges a signature.

E_3: $\mathcal{A}_1$ outputs a valid signature and C not terminating the game.

When all these events have occurred, the probability of success of C in the game is:

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1]P[E_2|E_1]P[E_3|E_2 \wedge E_1]$$

According to the definition of the game, we can have

- The probability of E_1 happening is at least $\left(1 - \frac{1}{(q_{PSK}+1)}\right)^{q_{PSK}}$. That is,

$$P[E_1] \geq \left(1 - \frac{1}{(q_{PSK} + 1)}\right)^{q_{PSK}}$$

- The probability of $E_2 \mid E_1$ happening is at least ϵ. That is,

$$P[E_2|E_1] \geq \epsilon$$

- The probability that $E_3 \mid E_2 \wedge E_1$ will occur is at least $\frac{1}{(q_{PSK}+1)}$. That is,

$$P[E_3|E_2 \wedge E_1] \geq \frac{1}{(q_{PSK}+1)}$$

Hence, we can have

$$\begin{aligned} P[E_1 \wedge E_2 \wedge E_3] &\geq \left(1 - \frac{1}{(q_{PSK}+1)}\right)^{q_{PSK}} \left(\frac{1}{(q_{PSK}+1)}\right)\epsilon \\ &\geq \left(1 - \frac{1}{(q_{PSK}+1)}\right)^{q_{PSK}} \left(1 - \frac{1}{(q_{PSK}+1)}\right)\frac{1}{q_{PSK}}\epsilon \\ &\geq \frac{1}{q_{PSK}}\left(1 - \frac{1}{(q_{PSK}+1)}\right)^{q_{PSK}+1}\epsilon \end{aligned}$$

For sufficiently large q_{PSK}, $\left(1 - \frac{1}{(q_{PSK}+1)}\right)^{q_{PSK}+1} \to \frac{1}{e}$. Therefore, C solves the CDH problem with a probability of $\frac{\epsilon}{q_{cu}e}$ within a time $t^{\ddagger} = t_c + t_m(q_{H_i} + q_{PSK} + q_{SK} + q_{PK} + q_{sig})$. ∎

Theorem 4. *The proposed scheme is existentially unforgeable against an adaptively chosen-message attack by a type II adversary in the random oracle model under the intractability of the CDH problem.*

Proof. Given a random instance $(P, X = aP, Y = bP)$ of the CDH problem, where $a, b \in \mathbb{Z}_q^*$ are unknown to $\mathcal{A}_2$. Suppose $\mathcal{A}_2$ has an advantage ϵ in forging a valid signature in the proposed scheme in time t_c, then there exists an algorithm which acts as a challenger C that can output $abP \in G_1$.

If $\mathcal{A}_2$ could forge a valid signature in our scheme by making $\{q_{H_i}\}_{i=1,2,3}$ queries to H_i-Queries, q_{SK} to Secret-Key-Queries, and q_{sig} to Sign-Queries; then, C can solve the CDH problem with a probability of at least $\frac{\epsilon}{q_{SK}e}$ within a time $t^{\ddagger} = t_c + t_m(q_{H_1} + q_{H_2} + q_{H_3} + q_{SK} + q_{sig})$, where e is the base of the natural logarithm and t_m is the time taken to perform a scalar multiplication operation in G_1. $\mathcal{A}_2$ interacts with C as follows:

Setup: C picks a challenged identity PID_i^* at random, sets $P_{pub} = X$, and submits the public parameters $params = \{q, G_1, G_2, e, P, P_{pub}, T_{pub}, H_0, H_1, H_2, H_3\}$ to $\mathcal{A}_2$. It should be noted that $\mathcal{A}_2$ does need to perform the Partial-Secret-Key-Queries because it holds the master secret key and it cannot make the Public-Key-Queries. As in Theorem 3, C also manages the lists L_{H_1}, L_{H_2}, and L_{H_3}. It also manages a list L_K containing the tuple (PID_i, vsk_i, c_i).

H_1-Queries, H_2-Queries, and H_3-Queries are the same as Theorem 3.

Secret-Key-Queries: On receiving a query on PID_i, C checks on the list L_K and performs the following.

- If the tuple $(PID_i, vsk_i, vpk_i, c_i)$ is found in L_K and $c_i = 0$, C terminates the game; otherwise, it returns vsk_i to $\mathcal{A}_2$.
- If the list L_K does not include the tuple $(PID_i, vsk_i, vpk_i, c_i)$, C checks the value of c_i. If $c_i = 0$, C sets $vpk_i = d_iP$, where $d_i \in \mathbb{Z}_q^*$; otherwise, it returns $vpk_i = d_iY \in G_1$. C sets $vsk_i = d_i$ and adds the tuple $(PID_i, vsk_i, vpk_i, c_i)$ to the list L_K. If $c_i = 1$, C returns vsk_i to $\mathcal{A}_2$; otherwise, it quits the game.

Sign-Queries: When $\mathcal{A}_2$ makes a query on (PID_i, m_i), C checks the lists L_{H_1}, L_{H_2}, and L_{H_3} to obtain the tuples $(ID_{R_j}, \alpha_j, H_j)$, $\{(m_i, PID_i, vpk_i, R_i, t_i, ID_{R_j}, h_i), (m_i, PID_i, r_i, vsk_i, \alpha_i)\}$, and (PID_i, μ_i, Q_i), respectively. **Forgery**: If $c_i = 1$ and $PID_i = PID_i^*$, $\mathcal{A}_2$ does not submit the sign query on (PID_i, vsk_i, vpk_i) to forge a signature $\sigma_i = (R_i, T_i)$ satisfying the equation

$$e(P, T_i) = e(P_{pub}, h_iQ_i)e(H_j, R_i + h_ivpk_i) \tag{4}$$

If Equation (4) holds, then C succeeds; otherwise, it fails.

According to Forking Lemma,[33] C can output another valid signature $\sigma_i^* = (R_i^*, T_i^*)$ satisfying the following equation:

$$e(P, T_i^*) = e(P_{pub}, h_i^*Q_i)e(H_j, R_i^* + h_i^*vpk_i) \tag{5}$$

We set $vpk_i = aQ_i = a\mu_i Y$, $Y = bP$, $P_{pub} = xP$, $H_j = \alpha_j P$. Now, we can have

$$e(H_j, R_i{}^* + h_i{}^* vpk_i) = e(P, T_i{}^*)e(P_{pub}, h_i{}^* Q_i)^{-1}$$

$$e(\alpha_j P, R_i{}^*)e(\alpha_j P, h_i{}^* \mu_i abP) = e(P, T_i{}^*)e(xP, h_i{}^* Q_i)^{-1}$$

$$e(\alpha_j P, h_i{}^* \mu_i abP) = e(P, T_i{}^*)e(xP, h_i{}^* Q_i)^{-1} e(\alpha_j P, R_i{}^*)^{-1}$$

$$\begin{aligned} e(\alpha_j h_i{}^* \mu_i abP, P) &= e(T_i{}^*, P)e((xh_i{}^* Q_i, P)e(\alpha_j R_i{}^*, P))^{-1} \\ &= e(T_i{}^* - (xh_i{}^* Q_i + \alpha_j R_i{}^*), P) \end{aligned}$$

Hence, we have

$$\alpha_j h_i{}^* \mu_i abP = T_i{}^* - (xh_i{}^* Q_i + \alpha_j R_i{}^*) \tag{6}$$

Therefore, C returns $abP = \{T_i{}^* - (xh_i{}^* Q_i + \alpha_j R_i{}^*)\}(\alpha_j h_i{}^* \mu_i)^{-1}$ as the solution to the give instance of CDH problem.
There are three events required for C to succeed in the game:
E_1: C does not abort the Secret-Key-Queries.
E_2: $\mathcal{A}_1$ forges a signature.
E_3: $\mathcal{A}_1$ outputs a valid signature and C not terminating the game.
When all these events have occurred, the probability of success of C in the game is:

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1]\mathrm{P}[E_2|E_1]P[E_3|E_2 \wedge E_1]$$

According to the definition of the game, we can have

- The probability of E_1 happening is at least $\left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}}$. That is,

$$P[E_1] \geq \left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}}$$

- The probability of $E_2 \mid E_1$ happening is at least ϵ. That is,

$$P[E_2|\mathrm{E}_1] \geq \epsilon$$

- The probability that $E_3 \mid E_2 \wedge E_1$ will occur is at least $\frac{1}{(q_{SK}+1)}$. That is,

$$P[E_3|E_2 \wedge E_1] \geq \frac{1}{(q_{SK}+1)}$$

Hence, we can have

$$\begin{aligned} P[E_1 \wedge E_2 \wedge E_3] &\geq \left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}} \left(\frac{1}{(q_{SK}+1)}\right)\epsilon \\ &\geq \left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}} \left(1 - \frac{1}{(q_{SK}+1)}\right)\frac{1}{q_{SK}}\epsilon \\ &\geq \frac{1}{q_{SK}}\left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}+1}\epsilon \end{aligned}$$

For sufficiently large q_{SK}, $\left(1 - \frac{1}{(q_{SK}+1)}\right)^{q_{SK}+1} \to \frac{1}{e}$. Therefore, C solves the CDH problem with a probability of $\frac{\epsilon}{q_{cu}e}$ within a time $t^{\ddagger} = t_c + t_m(q_{H_1} + q_{H_2} + q_{H_3} + q_{SK} + q_{sig})$. ■

TABLE 2 Computation cost comparison

Scheme	Sign (ms)	Verify (ms)	Aggregate verification (ms)	Secure
Zhong et al[5]	$3T_{p-m}+T_{p-a}+T_h=7.6163$	$3T_p+2T_{p-m}+T_{p-a}+T_H+T_h=27.2891$	$3T_p+2nT_{p-m}+(2n-1)T_{p-a}+nT_H+nT_h=(10.7843n+16.5041)$	No [Proposed]
Hashimoto and Ogata[23]	$3T_{p-m}+2T_{p-a}+2T_H=18.9939$	$4T_p+3T_H=39.0693$	$(n+3)T_p+(2n+1)T_H=(16.8652n+22.2041)$	Yes
Kumar et al[30]	$4T_{p-m}+2T_{p-a}+T_H+2T_h=15.8483$	$4T_p+3T_{p-m}+T_H+3T_h=35.3087$	$4T_p+3nT_{p-m}+(3n-3)T_{p-a}+T_H+(2n+1)T_h=(7.6604n+27.6483)$	No [Proposed]
Liu et al[31]	$3T_{p-m}+T_{p-a}+T_H+2T_h=13.2953$	$3T_p+2T_{p-m}+2T_{p-a}+2T_H+2T_h=32.9898$	$3T_p+2nT_{p-m}+(2n-1)T_{p-a}+(n+1)T_H+2nT_h=(10.7857n+22.1824)$	Yes
Proposed	$3T_{p-m}+T_{p-a}+2T_h=7.6170$	$3T_p+2T_{p-m}+T_{p-a}+T_H+T_h=27.2891$	$3T_p+2nT_{p-m}+(2n-1)T_{p-a}+nT_H+nT_h=(10.7843n+16.5041)$	Yes

7.2 | Performance evaluation

We present an extensive analysis of the computation cost and communication overhead of the proposed scheme and those in References 5,23,30,31, and then make comparison. The method of evaluation developed in Reference 34 is adopted in our analysis in which an 80-bit level of security is obtained using a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_T$, where G_1 is an additive group produced using a point P with prime order q on a super-singular elliptic curve $E: y^2 = x^3 + x \bmod p$ of embedding degree 2, where p and q are 512 bits and 160 bits, respectively. The experimental simulation is carried out on a personal computer running Windows 8, Intel I5-3320M 2.6 Hz processor, and a RAM of 4 GB using the standard MIRACL library.

7.2.1 | Computation overhead

We analyze the cost of computation in the sign, verify, and aggregate verify algorithms for all the five schemes. For clarity, we denote the time of performing a bilinear pairing, point multiplication, point addition, map-to-point hash function, and general hash function operations as T_p, T_{p-m}, T_{p-a}, T_H, and T_h, respectively. The running times of T_p, T_{p-m}, T_{p-a}, T_H, and T_h are 5.5086, 2.5313, 0.0217, 5.6783, and 0.0007 ms, respectively. The results of the computation cost of the proposed scheme and those in References 5,23,30,31 is shown in Table 2.

In Reference 5, a vehicle requires three point multiplication, one point addition, and one general hash function operations to sign a traffic message. So, the cost of computation of the Sign algorithm is $3 \times 2.5313 + 0.0217 + 0.0007 = 7.6163$ ms. To verify a message, a verifier needs to perform three bilinear pairing, two point multiplication, one point addition, one map-to-point hash function, and one general hash function operations. Hence, the computation cost is $3 \times 5.5086 + 2 \times 2.5313 + 0.0217 + 5.6783 + 0.0007 = 27.2891$ ms. An aggregate verifier in the scheme takes three bilinear pairing, $2n$ point multiplication, $(2n-1)$ point addition, n map-to-point hash function, and n general hash function operations. So, the cost of computation of the aggregate verify algorithm is $3 \times 5.5086 + 2n \times 2.5313 + (2n-1) \times 0.0217 + n \times 5.6783 + n \times 0.0007 = (10.7843n + 16.5041)$.

In the Sign algorithm of the proposed scheme, a verifier takes three point multiplication, one point addition, and two general hash function operations. Hence, the cost of computation is $3 \times 2.5313 + 0.0217 + 2 \times 0.0007 = 7.6170$ ms. The computation cost of the Verify and Aggregate Verify algorithms of the proposed scheme and[6] are the same. In Reference 23, a verifier executes three point multiplication, two point addition, and two map-to-point hash function operations. Hence, the cost of computation is $3 \times 2.5313 + 2 \times 0.0217 + 2 \times 0.0007 = 18.9939$ ms. A verifier needs four bilinear pairing and three map-to-point hash function operations to verify a traffic-related message. Thus, the computation cost is $4 \times 5.5086 + 3 \times 5.6783 = 39.0693$ ms. In the Aggregate Verify algorithm, an aggregate verifier takes $n+3$ bilinear pairing and $2n+1$ map-to-point hash function operations. Hence, the cost of computation is $(n+3) \times 5.5086 + (2n+1) \times 5.6783 = (16.8652n + 22.2041)$ ms.

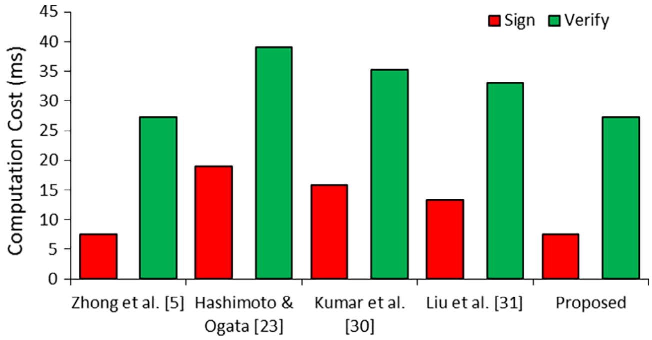

FIGURE 4 Computation cost of sign and verify

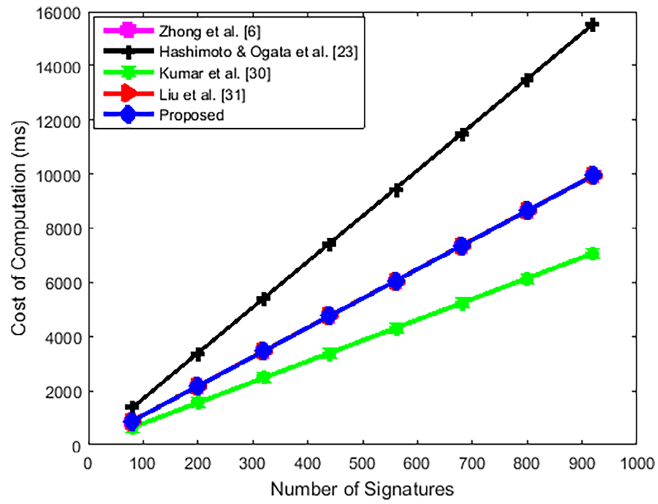

FIGURE 5 Computation cost of aggregate verify

The cost of computation of the schemes in References 30,31 can be analyzed in the same manner. The computation cost of the Sign and Verify stages for all the five schemes is shown in Figure 4. The cost of computation of the scheme in Reference 5 and the proposed scheme is the same and better than the other three schemes. However, we have demonstrated that the scheme in Reference 5 in insecure and therefore not suitable for practical deployment in VANET environment. The cost of computation in the Aggregate Verify phase of the five schemes with increase in the number of signatures is shown in Figure 5. Obviously, the cost increases linearly with the number of signers. The costs of aggregate verify stage of the proposed scheme,[5,31] overlap while that of Reference 30 slightly performs better than these three schemes. However, the scheme in Reference 5 is insecure against forgery attack and[30] is vulnerable to collusion attack.

7.2.2 | Communication overhead

We analyze the cost of transmitting a traffic-related information from a vehicle to the verifier in References 5,23,30,31, and the proposed scheme. Based on the construction of our scheme, the size of p is 64 bytes, hence the size of an element in G_1 is 128 bytes. We take the size of a timestamp and hash function output as 4 and 20 bytes, respectively. Since the size of traffic message is the same in all the schemes, we ignore it in our analysis.

In Reference 5 and the proposed scheme, a transmitted information from a vehicle to the verifier is $(PID_i, vPK_i, t_i, \sigma_i)$, where $PID_i = (PID_{i,1}, PID_{i,2}, VP_i)$, $\sigma_i = (R_i, T_i)$, $PID_{i,1}$, vPK_i, R_i, $T_i \in G_1$, $PID_{i,2} \in \mathbb{Z}_q^*$, VP_i is the validity period of PID_i, and t_i is the timestamp. So, the communication overhead of the proposed scheme and the one in Reference 5 is $128 \times 4 + 20 + 4 = 536$ bytes. A transmitted data from a vehicle to the verifier in Reference 23,31 is (ID_i, P_i, σ_i), where $\sigma_i = (R_i, S_i)$, $R_i, S_i, P_i \in G_1$, and $ID_i \in \mathbb{Z}_q^*$. Hence, the communication overhead in these two schemes is $128 \times 3 + 20 = 404$ bytes. In Reference 30, a vehicle submits a packet (PS_j, P_i, σ_i) to the verifier, where $PS_j = (PS1_j, PS2_j)$, $\sigma_i = (U_i, V_i)$, $PS1_j$, $PS2_j$, U_i, V_i, $P_i \in G_1$. Thus, the transmission overhead is $128 \times 5 = 640$ bytes. Table 3 shows the results of the communication overhead for all the five schemes. In Figure 6, we show the transmission overhead with increase in the number of signers. Although, the schemes in References 23,31 are slightly more efficiency than the proposed scheme in terms of

TABLE 3 Communication overhead comparison

Scheme	Single data (bytes)	*n* data (bytes)	Size of aggregate signature (bytes)
Zhong et al[5]	536	$536n$	$2\mid G_1\mid = 256$
Hashimoto and Ogata[23]	404	$404n$	$2\mid G_1\mid = 256$
Kumar et al[30]	640	$640n$	$n\mid G_1\mid + \mid G_1\mid = (128n + 128)$
Liu et al[31]	404	$404n$	$2\mid G_1\mid = 256$
Proposed	536	$536n$	$2\mid G_1\mid = 256$

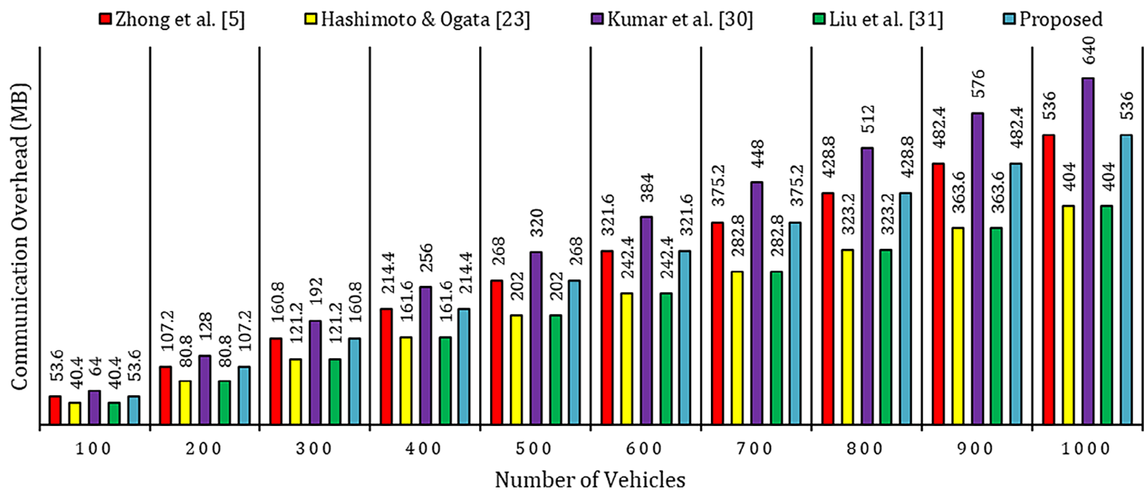

FIGURE 6 Communication overhead vs number of vehicles

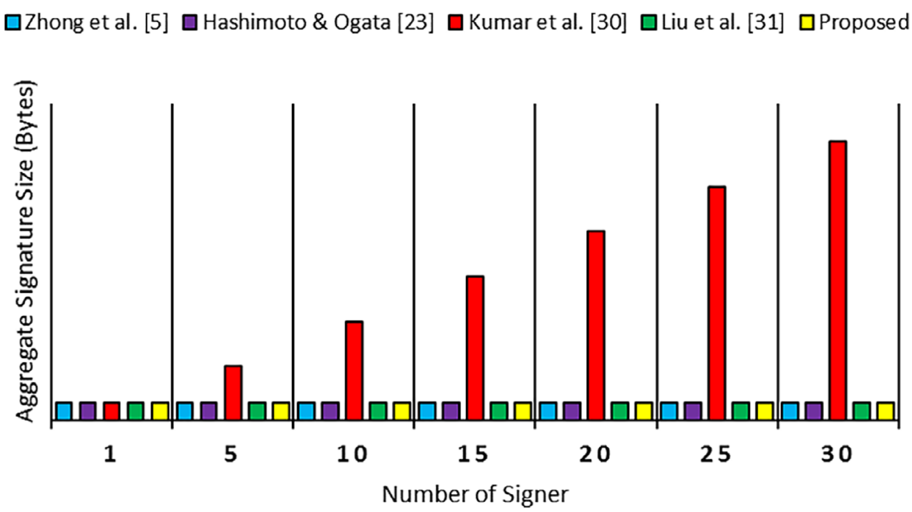

FIGURE 7 Aggregate signature size

communication overhead, the have shown that the scheme in Reference 23 cannot withstand replay attack. The size of the aggregate signatures with increase in signatures in the five schemes is shown in Figure 7. It is evident the aggregate signature size in the proposed scheme and those in References 5,23,31 is independent of the number of signatures while the size in Reference 30 increases linearly with the number of signatures.

8 | CONCLUSION

In this research article, we reviewed and analyzed Zhong et al's scheme[5] and found that the scheme is insecure under the type II adversary attack, thus making it unsuitable for practical deployment in Vehicular Ad Hoc Network environment. As a result, we proposed an improved scheme that can fix the weakness. We demonstrated that the improved scheme is secure against both type I and II adversaries in the random oracle model under the intractability of the Computational Diffie-Hellman Problem. Finally, we analyzed the efficiency of the new scheme and the related schemes. The results showed that the proposed scheme is superior to others.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

ORCID

Sunday O. Ogundoyin https://orcid.org/0000-0003-1564-2818

REFERENCES

1. Kenney JB. Dedicated short-range communications (DSRC) standards in the United States. *Proc IEEE*. 2011;99(7):1162-1182.
2. Ogundoyin SO. An autonomous lightweight conditional privacy-preserving authentication scheme with provable security for vehicular ad-hoc networks. *Int J Comput Appl*. 2018;1-16. https://doi.org/10.1080/1206212X.2018.1477320.
3. Ogundoyin SO. An efficient, secure and conditional privacy-preserving authentication scheme for vehicular ad-hoc networks. *J Inform Assur Secur*. 2017;12(5):179-192.
4. Kamil IA, Ogundoyin SO. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *J Inform Secur Appl*. 2019;44:184-200.
5. Zhong H, Shunshun H, Cui J, Zhang J, Xu Y. Privacy-preserving authentication scheme with full aggregation in VANET. *Inform Sci*. 2019;476:211-221.
6. Cui J, Zhang J, Zhong H, Shi R, Xu Y. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Inform Sci*. 2018;451-452:1-15.
7. De D, Zeadally S, Xu B, Huang X. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad-hoc networks. *IEEE Trans Inform Foren Secur*. 2015;10(12):2681-2691.
8. Isaac JT, Zeadally S, Camara J. Security attacks and solutions for vehicular ad-hoc networks. *IET Commmun J*. 2010;4(7):894-903.
9. Sari A, Onursal O, Akkaya M. Review of security issues in vehicular ad hoc networks (VANET). *Int J Commun Netw Syst Sci*. 2015;8:552-566.
10. Sucasas V, Mantas G, Saghezchi FB, Radwan A, Rodriguez J. An autonomous privacy-preserving authentication scheme for intelligent transportation systems. *Comput Secur*. 2016;60:193-205.
11. Raya M, Hubaux JP. The security of vehicular ad hoc networks. *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, Alexandria, VA*. New York: ACM; 2005:11-21.
12. Sun Y, Lu R, Lin X, Shen X, Su J. An efficient pseudonomous authentication scheme with strong privacy preservation for vehicular communications. *IEEE Trans Vehic Technol*. 2010;59(7):3589-3603.
13. Lu R, Lin X, Zhu H, Ho P, Shen X. ECPP: efficient conditional privacy preservation protocol for secure vehicular communications. *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, Phoenix, AZ, USA*. New York: IEEE; 2008. https://doi.org/10.1109/INFOCOM.2008.179.
14. Zhang C, Lin X, Lu R, Ho PH. RAISE: an efficient RSU-aided message authentication scheme in vehicular communication networks. *Proceedings of the IEEE international conference on communications (ICC'08), Beijing, China*. New York: IEEE; 2008:1451-1457.
15. Zhang C, Lin X, Ho PH, Shen X. An efficient identity-based batch verification scheme for vehicular sensor networks. *Proceedings of IEEE INFOCOM'08 - The 27th Conference on Computer Communications, Phoenix, AZ, USA*. New York: IEEE; 2008:816-824.
16. Lin X, Lu R. GSIS: group signature and ID-based signature-based secure and privacy-preserving protocol. *IEEE Vehicular Ad Hoc Network Security and Privacy*. Vol Chapter 2. New York: John Wiley & Sons Inc.; 2015:21-49.
17. Shim K. CPAS: an efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. *IEEE Trans Vehic Technol*. 2012;61(4):1874-1883.
18. Al-Riyami SS, Paterson KG. Certificateless public cryptography. *Proc. ASIACRYPT'03, LNCS*. Vol 2894. Taipei, Taiwan: LNCS; 2003:452-473.
19. Kang B, Wang M, Jing D. An efficient certificateless aggregate signature. *J Nat Sci*. 2017;22(2):165-170.

20. Qu Y, Mu Q. An efficient certificateless aggregate signature without pairing. *Int J Electron Secur Digital Foren*. 2018;10(2):188-203.
21. 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.
22. Yang X, Wang J, Ma T, Li Y, Wang C. A short certificateless aggregate signature against coalition attacks. *PLoS One*. 2018;13(12):e0205453.
23. Hashimoto K, Ogata W. Unrestricted and compact certificateless aggregate signature scheme. *Inform Sci*. 2019;487:97-114.
24. Kumar P, Kumari S, Sharma V, Sangaiah AK, Wei J, Li X. A certificateless aggregate signature scheme for healthcare wireless sensor networks. *Sustain Comput Inform Syst*. 2018;18:80-89.
25. Zhan Y, Wang B. Cryptanalysis of a certificateless aggregate signature scheme for healthcare wireless sensor network. *Secur Commun Netw*. 2019;2019:6059834. https://doi.org/10.1155/2019/6059834.
26. 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;2018:2595273. https://doi.org/10.1155/2018/2595273.
27. Xie Y, Li X, Zhang S, Li Y. iCLAS: an improved certificateless aggregate signature scheme for healthcare wireless sensor networks. *IEEE Access*. 2019;7:15170-15182.
28. Horng S, Tzeng S, Huang P, Wang X, Li T, Khan MK. An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks. *Inform Sci*. 2015;317:48-66.
29. Li J, Yuan H, Zhang Y. Cryptanalysis and improvement of certificateless signature with conditional privacy-preserving for vehicular sensor networks, 2016. IACR, Technical Report. https://eprint.iacr.org/2016/692. Accessed May 24, 2019.
30. 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. https://doi.org/10.1007/s11227-018-2312-y.
31. Liu J, Li Q, Cao H, Sun R, Du X, Guizani M. MDBV: monitoring data batch verification for survivability of internet of vehicles. *IEEE Access*. 2018;6:50974-50983.
32. Ogundoyin SO, Awoyemi SO. EDAS: efficient data aggregation scheme for internet of things. *J Appl Secur Res*. 2018;13(3):347-375.
33. David P, Jacque S. Security arguments for digital signatures and blind signatures. *J Cryptogr*. 2000;13(3):361-396.
34. Kamil IA, Ogundoyin SO. A lightweight CLAS scheme with complete aggregation for healthcare mobile crowdsensing. *Comput Commun*. 2019;147:209-224.

How to cite this article: Kamil IA, Ogundoyin SO. On the security of privacy-preserving authentication scheme with full aggregation in vehicular ad hoc network. *Security and Privacy*. 2020;3:e104.
https://doi.org/10.1002/spy2.104