Journal of Systems Architecture 99 (2019) 101636

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs

Ikram Ali, Mwitende Gervais, Emmanuel Ahene, Fagen Li*

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, PR China

ARTICLE INFO

Keywords:
Pseudo-identity
Privacy-preservation
Bilinear pairing
Certificateless signature
Un-linkability

ABSTRACT

Vehicular Ad Hoc Networks (VANETs) have been developing based on the state-of-art in wireless network communication technologies to improve traffic on roads. However, there are some threats to security and privacy due to the open wireless environment in VANETs and the high speed of vehicles. The authentication of messages related to traffic which are exchanged with the vehicles and the Road-Side Unit (RSU) is considered one of the most VANETs necessary security requirements. In this context, several schemes have been designed to secure the traffic-related messages in VANETs. However, these schemes suffer from high computational costs in signatures' verification. To minimize the computational cost of signature generation and verification, we propose an efficient Certificateless Public Key Signature (CL-PKS) scheme using bilinear pairing to provide conditional privacy-preserving authentication for Vehicle-To-Infrastructure (V2I) communication in VANETs. The CL-PKS scheme supports batch signature verification and aggregate signature verification functions to speed up verification process. In addition to this, we include blockchain to our CL-PKS scheme to implement revocation transparency of pseudo-identities efficiently before verifying the signatures. Furthermore, this scheme provides security proof and protection against different types of attacks. The proposed scheme incurs lower computational cost as compared to that incurred by existing schemes.

1. Introduction

An Intelligent Transportation System (ITS) performs an important role in traffic management on the road. The ITS provides pervasive and revolutionary services that will help better manage traffic caused by thousands of vehicles in metropolitan cities in the near future [1]. The fast-growing advancement in the state-of-the-art wireless communication technologies [2–4,33] allows manufacturers of vehicles and industries related to telecommunication to install wireless devices on vehicles. These permit vehicles to communicate traffic-related messages with each other and with infrastructure on the roadside (i.e. RSU) [5]. The network which supports this type of communication is known as VANET. An ordinary VANET environment consists of three fundamental network nodes, i.e. Trusted Authority (TA), RSU, and an On-Board Unit (OBU) mounted on the vehicle. The communication in VANETs is carried out through two ways such as Vehicle-To-Vehicle (V2V) and V2I. Fig. 1 depicts the overall scenario of VANETs. In V2V, a vehicle exchanges traffic-related messages with other vehicles through a Dedicated Short Range Communications (DSRC) protocol (IEEE 802.11p) within the coverage of the RSU [36,37]. In V2I, the vehicles and RSU or TA exchange traffic management information and infotainment services in order to improve the driving, journey, and safety of passengers.

However, with these advantages, the VANETs bring some issues and challenges related to the security and privacy of traffic-related messages [6–10,12]. The communications in VANETs are performed through an open wireless medium; the adversaries could use this to launch various kinds of attacks. For instance, the attacker can alter emergency type critical traffic-related messages that could spread across the VANETs to cause serious damages. Secure communications in VANETs manage traffic better and control traffic accidents, signal violations, traffic jam, parking difficulty, etc. Therefore, the traffic-related messages transmitted in VANETs must satisfy security requirements [17] such as authentication and integrity, conditional privacy preservation, un-linkability, etc. In these, authentication is the primary and essential one for effective VANETs security.

To secure the traffic-related messages in VANETs, some authentication schemes have been proposed using Public Key Infrastructure (PKI), Identity-based Cryptography (IDC) and CL Cryptography (CLC) focusing on different security requirements have been proposed. In the schemes based on PKI, a Certificate Authority (CA) keeps a huge number of certificates for managing vehicles public keys. As a result, the RSU requires sufficient computational power and storage capacity for the verification of these certificates on traffic-related messages. To alleviate this load caused from certificates' management from the schemes based on PKI,

* Corresponding author.
E-mail address: fagenli@uestc.edu.cn (F. Li).

https://doi.org/10.1016/j.sysarc.2019.101636
Received 20 May 2019; Received in revised form 4 July 2019; Accepted 21 August 2019
Available online 21 August 2019

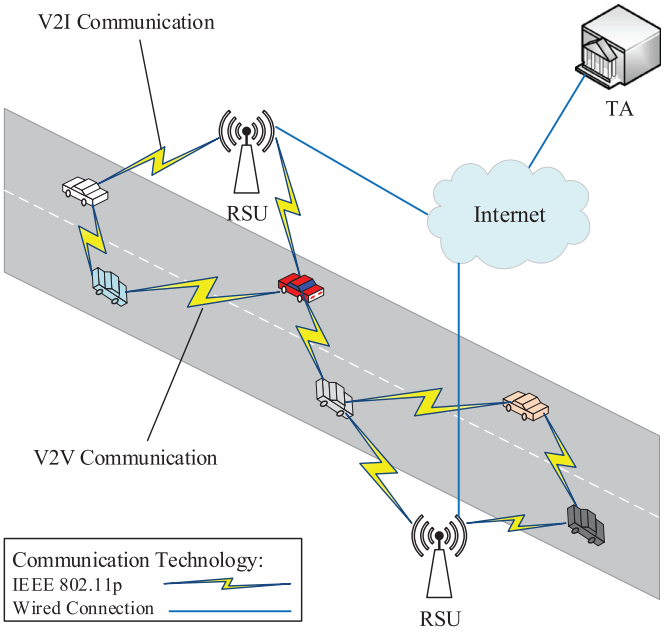

Fig. 1. A typical VANET scenario.

an IDC-based schemes are proposed. In IDC, the public key is derived from the identity information (i.e. email address, telephone number) of a user. However, in IDC, a Private Key Generator (PKG) can forge user's signature by using his private key, which is called key escrow problem. To remove this problem, Al-Riyami and Paterson proposed the concept of CLC in 2003 [18]. The researchers have also proposed the CL-Signature schemes [14,26–31] based on the CLC.

1.1. Motivations and contributions

The vehicle high-speed (i.e. greater than 36 km/h) and a maximum number of vehicles in VANETs challenge vehicular communication. The delay caused from the vehicles communications while moving with high speed must be extremely short to fulfil the severe timing requirement [13]. The aforementioned CL-Signature schemes [14,27–31] have been constructed using bilinear pairings over an elliptic curve. However, the most time-consuming cryptographic operation in signature verification on traffic-related message is the bilinear pairing operation over the elliptic curve. Its computational cost is larger than the other cryptographic operations' computational cost. Therefore, the bilinear pairing operation creates computational overhead while verifying the signature on traffic-related message.

The above problem motivates us to propose an efficient CL-PKS scheme using bilinear pairing to ensure conditional privacy-preserving authentication for V2I communication which is the most imperative need of the recent time for VANETs.

- First, we design an efficient CL-PKS scheme to minimize the number of pairing operations in signature verification to one pairing operation. We use general one-way hash functions instead of a special one-way hash function (map-to-point) to improve the performance. We also employ batch signature verification function to improve the performance further by verifying multiple signatures on traffic-related messages by the RSU at the same time. In addition to this, we use the aggregate signature verification function to enhance the bandwidth of the traffic-related communication between RSU and the Application Server.
- Second, we include blockchain [41] to introduce the revocation transparency efficiently in VANETs. This allows every network node in VANETs to check efficiently the revoked vehicles pseudo-identities with the Proof of Presence (PoP) and Proof of Absence (PoA) before verifying signatures on messages.
- Third, we conduct an exhaustive security analysis to illustrate that our proposed CL-PKS scheme fulfils requirements of security and privacy in V2I communication.
- Finally, the performance efficiency in terms of computation and communication costs is analysed to illustrate that our proposed CL-PKS scheme is efficient in comparison with the existing relevant CL-Signature schemes for vehicular communication.

1.2. Paper organization

The rest of the paper is organized as follows. In Section 2, the related work has been discussed. Section 3 describes an overview of blockchain technology briefly. The preliminaries for the proposed CL-PKS scheme have been discussed in Section 4. The construction of the proposed CL-PKS scheme is given with detail in Section 5. In Section 6, detailed security analysis has been given. The performance of proposed CL-PKS scheme is analysed comprehensively in Section 7. The application of proposed CL-PKS scheme in VANETs is provided in Section 8. Finally, Section 9 concludes the work of this paper.

2. Related work

In literature, many security and privacy schemes based on signature have been designed in order to guarantee authentication, integrity, privacy, non-repudiation, etc. of all traffic-related messages communicated in VANETs. In this domain, some authentication schemes based on PKI have been developed for VANETs. The signatures in these PKI schemes raise loads on the verifier vehicle in verifying signatures on traffic-related messages generated from different vehicles in VANETs. The OBU of verifier vehicle lack the capability to authenticate all signatures by verification in due time (i.e. 100–300 ms) due to the low computation power as a result computation cost is increased. Furthermore, the size of each transmitted packet is increased due to signature and public key certificate as a result communication cost is also increased.

To overcome the aforementioned problems, several schemes based IDC schemes have been designed to guarantee the authenticity and integrity of all messages transmitted in VANETs. The IDC has been introduced by Shamir [19] for the first time in 1984. IDC have solved various limitations in previous PKI-based schemes by removing certificates and correspondingly reduced the computation and communication costs.

However, in IDC, the PKG can use the users' private keys and forge signatures. To remove this problem, Al-Riyami and Paterson developed a mechanism of CLC in 2003 [18]. The researchers have also developed CL-Signature schemes based on the CLC. There are many application based CL-Signature schemes available in the literature [23–26]. However, it is not suitable for someone to use these schemes directly for authentication purpose in VANETs because every application has its own security requirements. Several CL-Signature schemes [14,27–31] have been proposed to offer authentication and integrity of messages transmitted in VANETs. Many of these schemes have been constructed from bilinear pairing and utilize signature aggregation function. In signature aggregation, all valid signatures on traffic-related message are aggregated by the RSU and further verified by the Application Server. However, it is also needed to verify multiple signatures at the same time instead of sequentially or aggregating them in VANETs, this type of verification is known as batch signature verification function. In 2014, Malip et al. [27] designed a novel CL-Signature scheme to authenticate and preserve the privacy of messages broadcasted in VANETS. However, under this scheme, the signature on traffic-related message is generated with three scalar multiplication and two map-to-point hash function operations while the signature on the message is verified with four bilinear pairing and three map-to-point hash function operations. In 2015, Malhi et al. [28] presented a new efficient CL aggregate signature scheme for the authentication of traffic-related message. However,

this scheme requires four scalar multiplication operations in message signing and three bilinear pairing and three scalar multiplication operations in signature verifying. Horng et al. [29] designed a conditional privacy preserving CL-Signature scheme for V2I communication. This scheme supports both batch and aggregate signature verification functions. However, the cost for the signature verification consists of three bilinear pairing, one scalar multiplication, and one map-to-point hash function operations which can affect vehicular communication. However, Li et al. [30] performed cryptanalysis and discussed the vulnerability to malicious but passive KGC attack on Horng et al.'s scheme [29] and proposed a secured scheme. However, their scheme needs two scalar multiplication and one map-to-point hash function operations to sign the traffic-related message and needs three bilinear pairing, one scalar multiplication, and two map-to-point hash function operations to verify the signature on message. In 2017, Tsai [14] designed a new CL short signature scheme based on bilinear pairing. This scheme has low computation costs in signing and verifying the message and is considered efficient among the existing schemes due to only one pairing operation. In 2018, Gayathri et al. [26] developed a bilinear pairing free efficient CL-Signature scheme for authentication of traffic-related messages in VANETs. This scheme improves the performance of VANETs with respective batch signature verification. Kumar et al. [31] designed a provable CL-Signature scheme based on bilinear pairing for VANETS. This scheme does not provide batch signature verification of traffic-related messages. However, the signing cost for the signature generation consists of four scalar multiplication and one map-to point hash function operations while the verifying cost for the signature consists of four bilinear pairing, three scalar multiplication, and one map-to-point hash function operations which can affect the performance of V2V and V2I communication in VANETs. Recently, Zhang et al. [22] addressed the issues related to data authenticity, third parties dishonesties, and system efficiency in cloud assisted Industrial Internet of Thing (IIoT). They designed a robust CL-Signature scheme which ensures data crowdsensing security in cloud assisted IIoT. This scheme provides provable security against both Type-I and Type-II attacks. The performance of this scheme is good because it requires only one scalar multiplication operation in signing process and one bilinear pairing operation and one scalar multiplication operation in verifying process.

Therefore, there is a demand for the design of those schemes which authenticate traffic-related messages with batch signature verification function efficiently in VANETs and provides provable security under the random oracle model.

3. An overview on blockchain technology

The blockchain technology was first introduced by Satoshi Nakamoto as the primary technology behind bitcoin protocol [41]. Bitcoin is called a first digital currency or crypto-currency or peer-to-peer e-cash system. The blockchain is a decentralized database that keeps a continuously increasing list of blocks containing all transactions. A transaction in a blockchain shows an action triggered by a participant. The blocks are chained to each other and are validated by the nodes take part in the peer-to-peer network [42]. The blockchain is a type of data structure which is hard in modifying because it contains the list of ordered data blocks linked by the cryptographic hash function (such as SHA-256) as shown in Fig. 2. Each block refers to a preceding block resulting in the blockchain. Once a block is made and inserted to the chain, then the transactions in that block cannot be modified and revised because the previous blocks are public and distributed as shown in Fig. 2. The blockchain as a distributed computational infrastructure allows a disruptive solution for the problem of double spending without the need of TA or CA in the middle to provide security and privacy when storing, monitoring, tracking, sharing and managing data [43]. The distributed ledger is encrypted by using Merkle tree or hash tree. The Merkle tree is a structure by which the leaf nodes are categorized with hashes of the data blocks and non-leaf nodes are categorized with their child nodes' cryptographic hashes [15]. It verifies the content of huge data structure securely and efficiently. It has also a consensus methodology which is based on the Proof of Work (PoW) algorithm. In this paper, we do not discuss in depth the implementation of Merkle tree in blockchain. We concisely describe here the following two data structures based on Merkel tree [16].

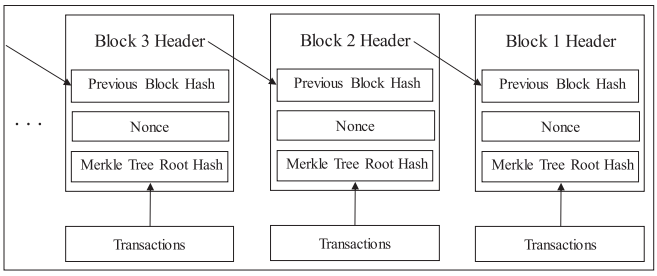

Fig. 2. Blockchain structure.

1. *Chronological Merkle Tree (CMT) and PoP*: It is considered the fundamental data structure for the design of blockchain in [21,44]. In CMT, the transactions from different nodes in peer-to-peer network are hashed in an order sequence. The root or top hash is then considered and incorporated into blockchain. Therefore the CMT generates and verifies the PoP for an issue, (i.e. un-revocation) efficiently. The detail about CMT is beyond the scope of this work. For detail see [16,41].
2. *Lexicographical Merkle Tree (LMT) and PoA*: The LMT rearranges all the information related to an issue, i.e. (revocation) into a single node of binary search tree efficiently. Therefore, LMT generates and verifies the PoA of such issue efficiently [45].

Presently, the researchers are trying to utilize blockchain technology in various fields, such as electronic voting, healthcare, Internet of Things, cloud computing [21], VANETs [16] and so on. In [21], Zhang et al. proposed a blockchain-based outsource service fair payment model called BPay in cloud computing without the involvement of a third party. Under this model, reliability and robust fairness are achieved. The advantage of BPay is that it is well-matched with the existing blockchains such as Bitcoin and Ethereum and is efficient with respect to computation cost. Lu et al. [16] designed a Blockchain-based Anonymous Reputation System (BARS) for VANETs. This system provides a privacy-preserving trust model in VANETs. The BARS uses blockchains connecting all the nodes in VANETs to efficiently ensure the transparency of certificates issued from CA. This also controls the transmission of false messages by reputation evaluation algorithm and enables vehicles to disclose the malicious behaviors of others.

4. Preliminaries for the proposed CL-PKS scheme

We briefly describe the bilinear pairing and the basic definitions of computationally hard problems, network model, framework, security notions and the use of blockchain for the proposed CL-PKS scheme.

4.1. Bilinear pairing and computational hard problems

Let G_1 and G_2 be a cyclic additive and cyclic multiplicative groups respectively having same prime order q, where as G_1 is generated by a point P. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing which satisfies the following properties [32,33].

- *Bilinearity*: $\forall\ a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(P, abP) = \hat{e}(abP, P)$.
- *Non-degeneracy*: There exists two points $P, Q \in G_1$, such that $\hat{e}(P, Q) \neq 1$ where 1 is the identity element in G_2.
- *Computability*: There must be an efficient algorithm which computes $\hat{e}(P, Q)\ \forall P, Q \in G_1$.

The following are some basic definitions of computationally hard problems which provide security foundation for our proposed CL-PKS scheme.

Definition 1. Discrete Logarithm (DL) problem: Given two points P and Q such that $P, Q \in G_1$, where P is a generator of G_1 having large prime order q. The task is to compute an integer a such that $Q = aP \in G_1$, where $a \in \mathbb{Z}_q^*$ is an unknown integer.

Definition 2. Computational Diffie–Hellman (CDH) problem: Given P, aP, $bP \in G_1$, where P is a generator of G_1 having large prime order q and $a, b \in \mathbb{Z}_q^*$, it is mathematically infeasible to compute $abP \in G_1$.

Definition 3. Collusion Attack Algorithm with k Traitors (k-CAA)[39]: For an integer k, and $x \in \mathbb{Z}_q^*$, $P \in G_1$, given $(P, Q = sP, h_1, h_2, \ldots, h_k \in \mathbb{Z}_q^*)$ and $((\frac{1}{h_1+s})P, (\frac{1}{h_2+s})P, \ldots, (\frac{1}{h_k+s})P)$, it is infeasible to compute $(\frac{1}{h+x})P$, where $h \notin (h_1, h_2, \ldots, h_k)$.

Definition 4. Inverse Computational Diffie–Hellman (*Inv*-CDH) Problem: Given P, $aP \in G_1$ for $a \in \mathbb{Z}_q^*$ and P is a generator of G_1 having large prime order q, it is mathematically infeasible to compute $(\frac{1}{a})P \in G_1$.

Inv-CDH problem is a hard problem because it is Probabilistic Polynomial Time (PPT) equivalent to CDH problem as mentioned above.

4.2. Network model

According to the [13], a VANET is organized into two layers. The lower layer consists of OBUs and RSUs, while the upper layer comprises TAs and Application Servers (traffic control centers). The construction of our proposed CL-PKS scheme is based on five network nodes such as two TAs: a Tracing Authority (TRA) and Key Generation Centre (KGC), Application Server, RSU, and OBU in vehicle as shown in the Fig. 8. The detail is described as follows.

1. **OBUs:** Vehicles are equipped with OBUs which are used to collect, compute and communicate traffic-related information through DSRC protocol (IEEE 802.11p) [36,37]. The OBU is a tamper-proof device (TPD) which never disclose his stored information. A clock is fixed in each OBU which can be securely resynchronized while moving in any RSU coverage. A Global Positioning System (GPS) and interface is also installed in each vehicle for location service and for interacting with drivers, respectively. The OBU computation power and storage capacity is less than RSU. Each vehicle signs traffic-related messages and transmits to the neighboring RSU.
2. **RSUs:** These are the base stations deployed along the roadside and act as intermediate nodes among OBUs, TRA, KGC, and Application Server. An RSU computation power and storage capacity is large than OBU and less than TA and KGC. Upon receiving the signed traffic-related messages, each RSU is responsible to verify the signatures on the messages. The RSUs then exchange these verified messages with the Application Server through a secure channel.
3. **Authentication server:** It is responsible to support applications which are used for safety purposes and traffic managements. It communicates with the RSUs to provide application support through a wired transport layer security protocol. It provides the authentic traffic-related information to each RSU after analysis. Each RSU then broadcasts such information within its own coverage in VANETs.
4. **TAs (TRA and KGC):** These TAs have the authority to manage and maintain VANETs. The TRA is responsible to register RSUs and OBUs. It generates pseudo-identities to provide identity anonymity among vehicles communications. It keeps secrets and takes the responsibility to resolve any dispute. It can trace and reveal the real identities of misbehaving entities (RSUs or vehicles) from signed messages and revokes them by revoking their registrations. While the KGC is responsible to create and assign partial private keys for OBUs. TRA and KGC have huge computation power and storage capacity than the RSU. In our proposed V2I communication model, it is assumed that the TRA and KGC are trusted network nodes due to strong security and do not conspired and compromised.

4.3. Framework

The generic construction of our proposed CL-PKS scheme comprises six algorithms: *Setup, PIDGen, PPKGen, SPKGen, CLSigGen*, and *CLSigVerify*.

1. ***Setup***: By taking a security parameter k, this algorithm responds with system public parameters *params* and master secret keys x and s for TRA and KGC, respectively. The TRA and KGC keep master secret keys safe.
2. ***PIDGen***: The TRA executes this algorithm that takes *params*, a real identity *RID*, and a partial pseudo-identity $PID_{i,1}$ from a vehicle V_i as input and outputs a pseudo-identity as PID_i to KGC and vehicle V_i.
3. ***PPKGen***: The KGC runs this algorithm that takes *params* and the pseudo-identity PID_i as input and responds with a partial private key PSK_i as output.
4. ***SPKGen***: The vehicle V_i with the pseudo-identity PID_i runs this algorithm by taking *params*, secret value β_i, partial private key PSK_i and responds with a private key SK_i and a concerned public key PK_i.
5. ***CLSigGen***: The vehicle V_i with the pseudo-identity PID_i runs this algorithm by taking *params*, message $M_i \in \{0, 1\}^*$ and private key SK_i as input and responds with a CL-Signature σ_i as output.
6. ***CLSigVerify***: A verifier executes this algorithm by taking *params*, message M_i with the PID_i, CL-Signature σ_i and public key PK_i and outputs true if σ_i is valid; otherwise outputs false. For simplicity, we do not consider both batch signature verification and aggregate signature verification functions here.

4.4. Security notions

In CLC, two adversaries types such as Type-I and Type-II with different capabilities are considered [18]. Type-I adversary considers an outside user as a malicious user who cannot access the master secret key of KGC, but can only replace adaptively the public key of any user by its chosen value. A Type-II adversary considers an honest but curious KGC who can access the master secret key of KGC but cannot replace the public key of any user by its chosen value. Furthermore, our proposed CL-PKS satisfies Existentially Un-Forgeability against an adaptively Chosen Message Attack (EUF-CMA) [34]. Therefore, for the proposed CL-PKS scheme, we assume two security notions, EUF-CMA-I for a Type-I adversary and EUF-CMA-II for a Type-II adversary. The two games that satisfy the two security notions are discussed as follows [35].

Game I: This is an un-forgeability game which is played between Type-I adversary $\mathcal{A}^I_{adv}$ and challenger C. In this game, C maintains a list to record the history of query-answer during interaction with $\mathcal{A}^I_{adv}$.

Phase 1-*Initial*: A challenger C executes the *Setup* that takes a security parameter k and sends the system public parameters to $\mathcal{A}^I_{adv}$ and keeps the master secret key with him/herself.

Phase 2-*Attack*: In this, $\mathcal{A}^I_{adv}$ performs the following polynomially bounded number of oracle-queries operations in an adaptively. It means that each query may depend on the result of the previous query.

- ***Generate partial-private key*:** $\mathcal{A}^I_{adv}$ submits a query (PID_i, *partial private key generate*) to C. On receiving this, C runs *PPKGen* algorithm to send the partial private key PPK_i to $\mathcal{A}^I_{adv}$. Note PID_i is not the challenged pseudo-identity.
- ***Generate private key*:** $\mathcal{A}^I_{adv}$ submits a query (PID_i, *private key generate*) to C. On receiving this, C runs *PPKGen* and *SPKGen* algorithms and sends the private key as SK_i to $\mathcal{A}^I_{adv}$.
- ***Set public key*:** $\mathcal{A}^I_{adv}$ submits a query (PID_i, *public key request*) to C. On receiving this query, C sets public key PK_i and sends it to $\mathcal{A}^I_{adv}$.
- ***Replace public key*:** $\mathcal{A}^I_{adv}$ with an pseudo-identity PID_i replace a public key PK_i by its chosen new secret value.

- ***CL-sign***: $\mathcal{A}^{I}_{adv}$ submits a query (PID_i, M_i, *signature*) to C. C finds private key SK_i from the list containing query-answer, runs *CLSigGen* algorithm and sends a signature σ_i to $\mathcal{A}^{I}_{adv}$. If the public key PK_i is replaced by $\mathcal{A}^{I}_{adv}$, then C cannot compute SK_i and thus the answer from the oracle may be wrong. Since, C does not know the sender secret value. In this situation, it is assumed that $\mathcal{A}^{I}_{adv}$ submits an additional secret value (which is related to the public key PK_i which he is replacing) to the *CL-sign* oracle.

Phase 3-*Forgery*: Finally, $\mathcal{A}^{I}_{adv}$ gives a CL-Signature σ^*_i on a message M^*_i corresponding to a challenged pseudo-identity PID^*_i and a public key PK^*_i.

$\mathcal{A}^{I}_{adv}$ wins the Game I if the following conditions hold.

- σ^*_i is a valid CL-Signature on the message M_i for the pseudo-identity PID^*_i and the corresponding public key PK^*_i.
- $\mathcal{A}^{I}_{adv}$ has never been queried the pseudo-identity PID^*_i to the *generate private key* oracle.
- $\mathcal{A}^{I}_{adv}$ has never been queried the pseudo-identity PID^*_i to both *generate partial-private key* and *replace public key* oracles.
- $\mathcal{A}^{I}_{adv}$ has never been queried the pseudo-identity PID^*_i with respect to message M^*_i and public key PK^*_i to the *CL-sign* oracle.

Definition 5. EUF-CMA-I: A CL-PKS scheme is existentially unforgeable against adaptive chosen message attack, if for any PPT Type-I adversary $\mathcal{A}^{I}_{adv}$, the success probability $Succ^{k}_{A_{adv}}$ is negligible, where $Succ^{k}_{A_{adv}}$ is the $\mathcal{A}^{I}_{adv}$ success probability to win the Game I.

Game II: This is an un-forgeability game which is played between Type-II adversary $\mathcal{A}^{II}_{adv}$ and challenger C. In this game, C maintains a list to record the history of query-answer during interaction with $\mathcal{A}^{II}_{adv}$.

Phase 1-*Initial*: A challenger C executes the *Setup* that takes a security parameter k and responds $\mathcal{A}^{II}_{adv}$ with the system public parameters and master secret key.

Phase 2-*Attack*: In this, $\mathcal{A}^{II}_{adv}$ performs polynomially bounded number of oracle-queries operations such as *generate private key, set public key*, and *CL-sign* which are similar to Game I.

Phase 3-*Forgery*: Finally, $\mathcal{A}^{II}_{adv}$ gives a CL-Signature σ^*_i on a message M^*_i for a challenged pseudo-identity PID^*_i and a corresponding public key PK^*_i.

$\mathcal{A}^{II}_{adv}$ wins the Game II if the following conditions hold.

- σ^*_i is a valid CL-Signature on the message M_i with the pseudo-identity PID^*_i and the corresponding public key PK^*_i.
- $\mathcal{A}^{II}_{adv}$ has never been queried the pseudo-identity PID^*_i to the *generate private key* oracle.
- $\mathcal{A}^{II}_{adv}$ has never been queried the message M^*_i for the pseudo-identity PID^*_i with respect to public key PK^*_i to the *CL-sign* oracle.

Definition 6. EUF-CMA-II: A CL-PKS scheme is existentially unforgeable against adaptive chosen message attack, if for any PPT Type-II adversary $\mathcal{A}^{II}_{adv}$, the success probability $Succ^{k}_{A_{adv}}$ is negligible, where $Succ^{k}_{A_{adv}}$ is the $\mathcal{A}^{II}_{adv}$ success probability to win the Game II.

4.5. Blockchain in proposed CL-PKS scheme

In our proposed scheme, there are two blockchains to which the network nodes such as TRA, KGC, RSU, and vehicles are connected as shown in Fig. 3.

1. *Blockchain for Pseudo-Identities (PID-BC)*: TRA generates PID-BC which works as the public database for all those pseudo-identities which have generated but have not been revoked by TRA. It provides PoP of the pseudo-identity of sender vehicle for the verifier before verifying signature on message with $O(\log^N)$ efficiency, where N denotes the number of leaves in the Merkle tree.
2. *Blockchain for Revoked Pseudo-Identities (RPID-BC)*: TRA generates RPID-BC which works as the public database for all those pseudo-identities which have been revoked by TRA upon the request of the verifier. It offers PoA of the pseudo-identity of sender vehicle for the verifier before verifying signature on message with $O(\log^N)$ efficiency, where N denotes the number of leaves in the Merkle tree.

The pseudo-identities in PID-BC are stored chronologically in CMTs. Therefore, PID-BC is analogous to the conventional blockchain in bitcoin. The RPID-BC comprises both CMT and LMT. The vehicles with revoked pseudo-identities are stored chronologically in the LMT and whose root is changed when a new vehicle with revoked pseudo-identity is added in it. The transaction about pseudo-identity revocation and the concerned LMT root is stored chronologically by CMT. The CMT transaction root and the LMT pseudo-identity root are saved in the block header [16]. The discussion on consensus mechanism based on the PoW offered by vehicles is beyond the scope of this work.

In this paper, we conceptually discuss the blockchain inclusion in VANETs and thereby this does not answer all the questions about blockchain.

5. Proposed CL-PKS scheme

We design a CL-PKS scheme based on blockchain to improve performance with respect to computational cost of V2I communication in VANETs. Under the proposed CL-PKS scheme, we improve the performance efficiency of the scheme [30] to verify multiple signatures by utilizing the batch signature verification function. In this scheme, two blockchains (i.e. PID-BC and RPID-BC) are used which connect network nodes (such as TRA, KGC, RSU, and vehicles) through the Internet as shown in Fig. 3. The notations used in this paper are given in Table 1.

5.1. Construction

The following is the detailed construction of the proposed CL-PKS scheme.

1. ***Setup***

   This algorithm is run by TRA and KGC which takes a security parameter 1^k for $k \in N$ and generates the parameters first as (G_1, G_2, e), where $(G_1, +)$ and $(G_2, \cdot)$ are two cyclic groups having large prime order q and e: $G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing, as described in Section 4.

   1. Let TRA and KGC choose a random generator P of G_1 and let $g = \hat{e}(P, P)$.
   2. TRA chooses a random number $x \in \mathbb{Z}^*_q$ as its master secret key and sets $T_{pub} = xP$ as its master public key.
   3. KGC selects $s \in \mathbb{Z}^*_q$ randomly as its master secret key and sets $P_{pub} = (P_{pub1}, P_{pub2}) = (sP, (\frac{1}{s})P)$ as its master public key.
   4. TRA and KGC choose three distinct cryptographic general hash functions H_0, H_1, and H_2 as H_0: $\{0, 1\}^* \rightarrow \mathbb{Z}^*_q$, $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}^*_q$, and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}^*_q$.
   5. TRA and KGC publish system parameters publicly as *params*= (p, q, P, $G_1, G_2, \hat{e}, g, T_{pub}, P_{pub}, H_1, H_2$). The TRA and KGC keep master secret keys x and s safe, respectively. Notice H_0 hash function is used only by TRA for pseudo-identity generation, therefore, it is not published.
2. ***PIDGen***

   Upon the request of a vehicle V_i, TRA generates a pseudo-identity PID_i for the vehicle V_i. This algorithm works as follows.

   1. Vehicle V_i first selects a number $\alpha_i \in \mathbb{Z}^*_q$ randomly, computes $PID_{i,1} = \alpha_i P$ and sends (RID, $PID_{i,1}$) to TRA via a secure channel, where as RID is the real identity of the vehicle V_i.
   2. After checking the uniqueness of RID of vehicle V_i, TRA computes $PID_{i,2} = RID \oplus H_0(xPID_{i,1}, T_i)$, where as x is the TRA's secret key and T_i is the time-stamp that shows the period validity.
   3. TRA sets $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$ as the pseudo-identity for the vehicle V_i and sends it to KGC and vehicle V_i via the secure channel. TRA then signs PID_i using its master secret key x and inserts it in the PID-BC.

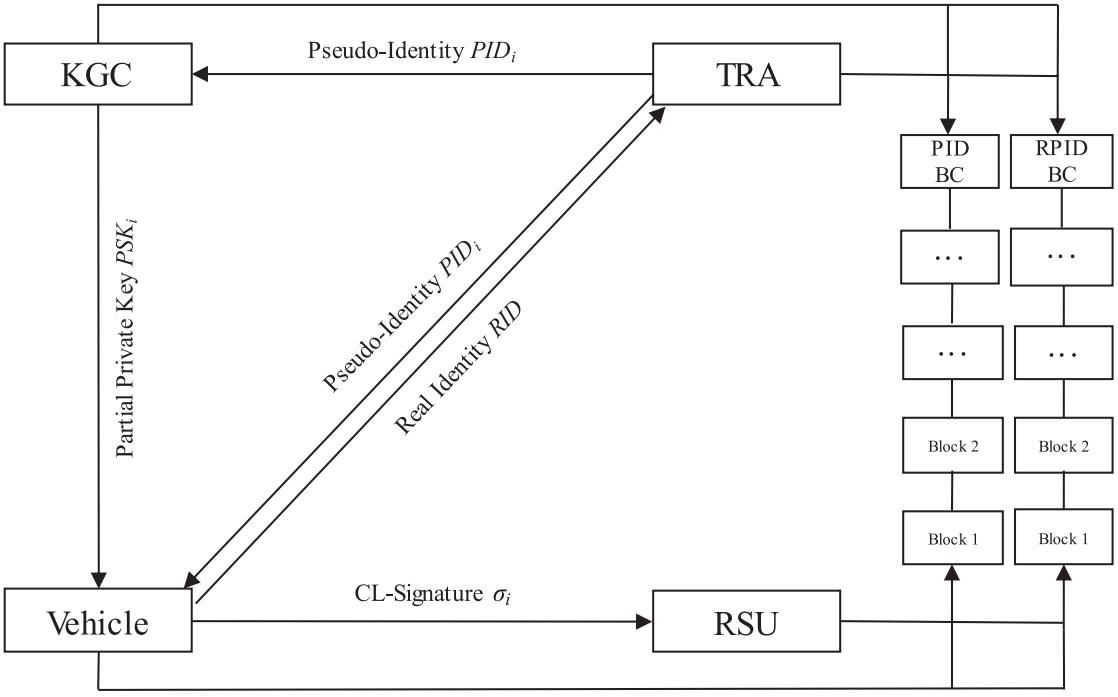

Fig. 3. Proposed scheme.

3. ***PPKGen***
Once received the pseudo-identity PID_i from TRA, KGC works as follows.
1. KGC first looks up the PID-BC and RPID-BC using T_{pub} to ensure that PID_i is present in PID-BC and absent in RPID-BC. Which means that PID_i has been assigned to vehicle V_i and has not been revoked by TRA. CMT and LMT in PID-BC and RPID-BC present the PoP and PoA of PID_i for the KGC with $O(\log^N)$ efficiency. The KGC then generates a partial private key PSK_i for the vehicle V_i as follows.
2. KGC chooses $\gamma_i \in \mathbb{Z}_q^*$ randomly, computes $A_i = \gamma_i P$, $h_i = H_1(PID_i, A_i, P_{pub})$ and $\theta_i = (s + h_i\gamma_i) \bmod q$, where as s is the KGC's master secret key.
3. KGC sets $PSK_i = (\theta_i, A_i)$ as the partial private key and sends it to vehicle V_i via the secure channel.
4. ***SPKGen***
Upon receiving the partial private key PSK_i from KGC and pseudo-identity PID_i from TRA, the vehicle V_i works as follows.
1. Vehicle V_i first authenticates the partial private key PSK_i by verifying the equation $\theta_i P = P_{pub1} + h_i A_i$ as.

Table 1
Definitions of different notations.

Notation	Description
TRA	A tracing authority
KGC	A key generation center
RSU	A road-side unit
OBU	An On-Board Unit
V_i	An ith vehicle
G_1	An additive cyclic group
P	A generator point of G_1
G_2	A multiplicative cyclic group
q	An order of G_1 and G_2
$\hat{e} : G_1 \times G_1 \rightarrow G_2$	A bilinear pairing
x	TRA's master secret key
T_{pub}	TRA's master public key
s	KGC's master secret key
P_{pub}	KGC's master public key
RID_i	Vehicle V_i real identity
$PID_{i,1}$	A part of pseudo-identity generated by V_i
$PID_{i,2}$	A part of pseudo-identity generated by TRA
$PID_i = (PID_{i,1}, PID_{i,2})$	Vehicle V_i pseudo-identity generated by TRA
t_i, T_i	A valid time periods
PSK_i	Vehicle V_i partial private key
β_i	A secret key chosen by vehicle V_i
SK_i	Vehicle V_i private key
PK_i	Vehicle V_i public key
$M_i \in \{0, 1\}^*$	A message generated by vehicle V_i
$H_0(\cdot)$, $H_1(\cdot)$, and $H_2(\cdot)$	Three one-way general hash functions take $\{0,1\}^* \rightarrow \mathbb{Z}_q^*$
$\oplus$	An exclusive-OR operation
σ_i	A signature on M_i

Proof of correctness:

$$\begin{aligned}\theta_i P &= (s + h_i\gamma_i)P \\ &= sP + h_i\gamma_i P \\ &= P_{pub1} + h_i A_i\end{aligned}$$

The vehicle V_i does not accept the partial private key PSK_i if the above equation does not hold; otherwise computes as follows.

2. Vehicle V_i then selects a secret value $\beta_i \in \mathbb{Z}_q^*$ randomly and sets the $SK_i = (\beta_i, \theta_i)$ as its private key.

3. Vehicle V_i computes $U_i = \beta_i P_{pub1}$, $R_i = \theta_i P_{pub1}$ and sets $PK_i = (R_i, U_i)$ as its public key and broadcasts it in VANETs.

We recommend a preloading method [11] for CL-PKS scheme to achieve different and unique pseudo-identity each time and therefore preserve the privacy of the vehicle. A group of pseudo-identities having short expiration times with different changing intervals and partial private keys are loaded into the vehicle via TAs after running *PIDGen* and *SPKGen* algorithms. When all the pseudo-identities with the vehicle are running out and the vehicle communication to TAs after proper authentication with suitable bandwidth is established in VANETs. Then the vehicle pseudo-identities are replenished through a secure medium between the concerned vehicle and TAs.

5. ***CLSigGen***

On inputs *params*, message $M_i \in \{0, 1\}^*$, pseudo-identity PID_i, Private key SK_i, and public key PK_i to this algorithm.

1. Vehicle V_i computes $j_i = H_2(M_i, PID_i, A_i, PK_i, P_{pub}, t_i)$ and generates the following CL-Signature as.

$$\sigma_i = \left(\frac{1}{j_i\theta_i + \beta_i}\right) P_{pub2} \tag{1}$$

σ_i is a CL-Signature on the message M_i for the pseudo-identity PID_i within a fixed interval of time t_i.

2. Vehicle V_i broadcasts the entire message-signature tuple (M_i, PID_i, Pk_i, σ_i, t_i) to a nearby verifier, i.e. RSU.

The steps above after *Setup* algorithm are repeated every 100–300 ms due to the DSRC protocol [36,37].

6. ***CLSigVerify***

Upon receiving the traffic-related message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) signed by vehicle V_i, a concerned verifier has the responsibility to verify the signature σ_i on message M_i to guarantee that the sender vehicle V_i is not trying to pretend to be any other authorized vehicle or broadcasts bogus messages.

1. RSU first validates the message M_i by the checking the timestamps T_i in the pseudo-identity PID_i and t_i in the message-signature tuple, whether these are in a valid time intervals. If hold then RSU can continue checking further.

2. RSU then looks up PID-BC and the RPID-BC for the pseudo-identity PID_i using T_{pub} to ensure that the received PID_i is present in PID-BC and absent in RPID-BC. Which means that PID_i has been assigned to vehicle V_i and has not been revoked by TRA. CMT and LMT in PID-BC and RPID-BC present the PoP and PoA of the PID_i for RSU with $O(\log^N)$ efficiency. If this holds then verifier can continue the verification further.

In this scheme, the RSU verifies the authenticity and integrity of traffic-related message M_i generated by vehicle V_i by two functions *single signature verification* and *batch signature verification*, which are described as follows.

(a) ***Single signature verification***

Through this type of verification, signatures on messages are verified one after the other sequentially.

Once the RSU receives a message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) sent by the vehicle V_i and after checking the timestamps (i.e. T_i in the pseudo-identity PID_i and t_i in the message-signature tuple) and pseudo-identity PID_i (i.e. in PID-BC and RPID-BC), the RSU utilizes *params*, M_i, PID_i, PK_i, and σ_i, computes $h_i = H_1(PID_i, A_i, P_{pub})$ and $j_i = H_2(M_i, PID_i, A_i, PK_i, P_{pub}, t_i)$ and checks whether the following equation is verified or not.

$$\hat{e}(\sigma_i, j_i R_i + U_i) = g \tag{2}$$

If holds, then RSU accepts the CL-Signature σ_i on message M_i otherwise rejects it.

Proof of correctness:

Taking L.H.S of Eq. (2)

$$\begin{aligned}\hat{e}(\sigma_i, j_i R_i + U_i) &= e\left(\left(\frac{1}{j_i\theta_i + \beta_i}\right) P_{pub2}, j_i R_i + U_i\right) \\ &= \hat{e}\left(\left(\frac{1}{j_i(s + h_i\gamma_i) + \beta_i}\right) P_{pub2}, j_i\theta_i P_{pub1} + \beta_i P_{pub1}\right) \\ &= \hat{e}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}\right) P, j_i(s + h_i\gamma_i)sP + \beta_i sP\right) \\ &= \hat{e}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}\right) P, (j_i s + j_i h_i\gamma_i + \beta_i)sP\right) \\ &= \hat{e}(P, P)^{\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}(j_i s + j_i h_i\gamma_i + \beta_i)s} \\ &= \hat{e}(P, P) \\ &= g = R.H.S\end{aligned}$$

Thus, the correctness of single signature verification proved. The right hand side g of Eq. (2) is already computed in the *Setup* algorithm. In single verifying n signature on n message, the RSU would need n pairing operation, where $n = 1, 2, 3, \ldots$ ..

(b) ***Batch signature verification***

Through batch signature verification function, multiple signatures on multiple traffic-related messages generated from different vehicles are verified by a single verifier simultaneously which save the number of pairing operations and makes efficient the overall performance of V2I communication in VANETs.

Once the RSU receives multiple distinct message-signature tuples ($PID_1, M_1, PK_1, \sigma_1, t_1$), ($PID_2, M_2, PK_2, \sigma_2, t_2$), ...,($PID_n, M_n, PK_n, \sigma_n, t_n$) for $i = 1, 2, 3, \ldots ., n$ from different vehicles ($V_1, V_2, \ldots .V_n$) and after checking the timestamps (i.e. T_i in the pseudo-identity PID_i and t_i in the message-signature tuple) and pseudo-identity PID_i (i.e. in PID-BC and RPID-BC), the RSU utilizes *params*, M_i, PID_i, PK_i, and σ_i, computes $h_i = H_1(PID_i, A_i, P_{pub})$ and $j_i = H_2(M_i, PID_i, A_i, PK_i, P_{pub}, t_i)$ and checks whether the following equation is verified or not.

$$\hat{e}\left(\sum_{i=1}^{n}(\sigma_i, j_i R_i + U_i)\right) = g \tag{3}$$

If holds, then verifier accepts the CL-Signature σ_i on message M_i otherwise rejects it.

Proof of correctness:

Taking L.H.S of Eq. (3)

$$\begin{aligned}&\hat{e}\left(\sum_{i=1}^{n}(\sigma_i, j_i R_i + U_i)\right) \\ &= \hat{e}\left(\sum_{i=1}^{n}\left(\left(\frac{1}{j_i\theta_i + \beta_i}\right) P_{pub2}, j_i R_i + U_i\right)\right) \\ &= \hat{e}\left(\sum_{i=1}^{n}\left(\left(\frac{1}{(j_i(s + h_i\gamma_i) + \beta_i}\right) P_{pub2}, j_i\theta_i P_{pub1} + \beta_i P_{pub1}\right)\right) \\ &= \hat{e}\left(\sum_{i=1}^{n}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}\right) P, j_i(s + h_i\gamma_i)sP + \beta_i sP\right)\right) \\ &= \hat{e}\left(\sum_{i=1}^{n}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}\right) P, (j_i s + j_i h_i\gamma_i + \beta_i)sP\right)\right) \\ &= \hat{e}(P, P)^{\sum_{i=1}^{n}\left(\frac{1}{(j_i s + j_i h_i\gamma_i + \beta_i)s}(j_i s + j_i h_i\gamma_i + \beta_i)s\right)} \\ &= \hat{e}(P, P) \\ &= g = R.H.S\end{aligned}$$

Thus, the batch signature correctness is verified. The right hand-side g of Eq. (3) is already computed in the *Setup* algorithm. Therefore, through batch signature verification function, the RSU requires only one pairing operation to verify n signature on n message at the same time.

5.2. CL-PKS aggregation

To get benefit from the verifier computation power and improve the CL-Signature verification further, we use an aggregate signature verification function which has been introduced by Boneh et al. [38]. Given n CL-Signature on n message generated from n sender, the verifier collects all these and arranges them together into a single one. The resultant aggregate signature makes the verifier certain that n vehicle has signed the n corresponding traffic-related message. This characteristic significantly decreases the computational cost in CL-Signatures verification and the storage cost in transmission of CL-Signatures.

The proposed CL-Signature aggregation scheme based on the aforementioned proposed CL-PKS scheme. The algorithms (i.e. *Setup, PIDGen, PPKGen, SPKGen, CLSigGen* are used for the proposed CL-Signature aggregation scheme. In addition to this, we consider the following two algorithms such as *ACLSigGen* and *ACLSigVerify*.

1. ***ACLSigGen***
   Each RSU performs as aggregate signature generator who aggregates a bundle of CL-Signatures into single one, once received the verified messages-signatures tuples (M_1, PID_1, PK_1, σ_1, t_1), (M_2, PID_2, PK_2, σ_2, t_2),....,(M_n, PID_n, PK_n, σ_n, t_n) generated from n vehicle ($V_1, V_2, \dots, V_n$). To aggregate a bundle of the CL-Signatures ($\sigma_1, \sigma_2, \dots, \sigma_n$) on messages ($M_1, M_2, \dots, M_n$) for the pseudo-identities ($PID_1, PID_2, \dots, PID_n$) and the corresponding public keys ($PK_1, PK_2, \dots, PK_n$), RSU computes $\sigma_{agg} = \sum_{i=1}^{n} \sigma_i$, outputs a CL aggregate signature as $\sigma_{agg} = (\sigma_1, \sigma_2, \dots, \sigma_n)$ and sends it to Application Server.
2. ***ACLSigVerify***
   Upon receiving the CL aggregate signature $\sigma_{agg} = (\sigma_1, \sigma_2, \dots, \sigma_n)$ on messages ($M_1, M_2, \dots, M_n$) for pseudo-identities ($PID_1, PID_2, \dots, PID_n$) and the corresponding public keys ($PK_1, PK_2, \dots, PK_n$) sent from n vehicle ($V_1, V_2, \dots, V_n$).
   - The Application Server first checks the validity of T_i and t_1, for $i = 1, 2, \dots, n$, if these hold, Application Server then utilizes *params*, M_i, PID_i, PK_i, and $\sigma_{agg} = (\sigma_1, \sigma_2, \dots, \sigma_n)$.
   - The Application Server then computes $j_i = H_2(M_i, PID_i, A_i, PK_i, P_{pub}, t_i)$, for $i = 1, 2, \dots, n$ to check whether the following equation is verified or not.

$$\hat{e}\left(\sum_{i=1}^{n}(\sigma_{agg}, j_i R_i + U_i)\right) = g \tag{4}$$

   The Application Server accepts the CL aggregate signature if the above equation holds; otherwise rejects it.

6. Security analysis

In this section, our proposed CL-PKS scheme's security is proved in the random oracle model and analysed its security requirements.

6.1. Security proof

Based on the following theorems, we prove that the proposed CL-PKS scheme is un-forgeable against Type-I adversary and Type-II adversary in the random oracle model. The hard problems in CL-PKS are similar to the hard problems in Du and Wen's scheme [40].

Theorem 1. *In the random oracle model, our CL-PKS scheme is EUF-CMA secure against Type-I adversary and Type-II adversary with the assumptions that $k-$CAA problem in G_1 and Inv-CDH problem in G_1 are intractable.*

The proof of this theorem according to the Definitions 5 and 6 follows the following Lemmas 1 and 2.

Lemma 1. *If an adversary $\mathcal{A}^I_{adv}$ with non-negligible advantage ε against the EUF-CMA-I security of proposed CL-PKS scheme running in time t and performing q_{H_1} and q_{H_2} queries to H_1 and H_2 random oracles and q_{psk}, q_{sk}, q_{pk}, and q_{sig} queries to the generate partial-private key, generate private key, set public key, and CL-sign oracles, then a challenger C_1 is constructed to solve the $k-$CAA problem with the following advantage.*

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

within time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$, where t_{sm} is the time needed for the computation of a scalar multiplication in G_1 and t_{inv} is the time needed for the computation of inversion in G_1.

Proof of Theorem. We consider an adversary $\mathcal{A}^I_{adv}$ of Type-I which breaks our proposed CL-PKS scheme. We consider a challenger C_1 to exploit $\mathcal{A}^I_{adv}$ for solving the $k-$CAA problem in G_1. $\mathcal{A}^I_{adv}$ and C_1 perform the adversary and challenger role respectively. C_1 accepts a challenge which contains random instance of $k-$CAA problem, i.e. given P, $P_{pub} = (P_{pub1}, P_{pub2}) = (sP, s^{-1}P) \in G_1$, $h_1, h_2, \dots, h_{q_{psk}}$, and $(\frac{1}{s+h_1})P_{pub1}, (\frac{1}{s+h_2})P_{pub1}, \dots, (\frac{1}{s+h_{q_{psk}}})P_{pub1}$, the task of C_1 is to compute $(h^*, (\frac{1}{j^*(s+h_i^*\gamma_i^*)+\beta_i^*})P_{pub1})$, where $h^* \notin (h_1, h_2, \dots, h_{q_{psk}})$. We consider two random oracles (i.e. H_1 and H_2) which are queried by the $\mathcal{A}^I_{adv}$. We also assume that a hash query for the pseudo-identity PID_i to H_1 oracle is performed before the keys extraction or signing queries for that pseudo-identity PID_i. C_1 keeps two hash lists L^{H_1} and L^{H_2} and a public key list L^{pk} which are initially empty. C_1 interacts with $\mathcal{A}^I_{adv}$, responds to all the queries of $\mathcal{A}^I_{adv}$, and records these in the lists L^{H_1} and L^{H_2} and L^{pk} throughout Game I which are as follows.

- *Setup*: This algorithm is run by C_1 which takes a security parameter 1^k for $k \in N$ and responds with system public parameters $params = (G_1, G_2, P, g, P_{pub}, H_1, H_2)$ and master secret key $s \in \mathbb{Z}_q^*$, where $g = \hat{e}(P, P)$ and $P_{pub} = (P_{pub1}, P_{pub2}) = (sP, s^{-1}P)$. Note the secret key s is unknown to C_1. C_1 selects a pseudo-identity PID_i^* randomly as a challenged pseudo-identity for $\mathcal{A}^I_{adv}$ in this game and sends the *params* to $\mathcal{A}^I_{adv}$.
- H_1 Oracle: When $\mathcal{A}^I_{adv}$ with a pseudo-identity PID_i performs this query to H_1 oracle, C_1 checks whether the list L^{H_1} comprises (PID_i, A_i, h_i) for the PID_i. If it does, C_1 responds $\mathcal{A}^I_{adv}$ with a previously defined H_1 hash value; if it does not, C_1 performs as follows.
  - If $PID_i = PID_i^*$ then C_1 responds $\mathcal{A}^I_{adv}$ with a random $h' \notin (h_1, h_2, \dots, h_{q_{psk}})$ hash value to $\mathcal{A}^I_{adv}$.
  - If not, C_1 chooses a value as $h_i \in (h_1, h_2, \dots, h_{q_{psk}})$ randomly and returns this to $\mathcal{A}^I_{adv}$. C_1 then inserts (PID_i, A_i, h_i) into the list L^{H_1}.
- *Generate partial-private key*: If $\mathcal{A}^I_{adv}$ with a pseudo-identity PID_i executes this query, C_1 performs as follows.
  - If $PID_i \neq PID_i^*$, C_1 selects two random numbers $u_i \in \mathbb{Z}_q^*$ and $h_i \in \mathbb{Z}_q^*$, computes $A_i = h_i^{-1}(u_i P - P_{pub1})$ and sets $\theta_i = u_i$ and $H_1(PID_i, A_i, P_{pub}) = h_i$. C_1 then sets the partial private key as $PSK_i = (\theta_i, A_i)$, sends it to $\mathcal{A}^I_{adv}$, and inserts (PID_i, A_i, h_i) into the list L^{H_1}.
  - If $PID_i = PID_i^*$, C_1 then halts and outputs "failure".
- *Set public key*: Suppose $\mathcal{A}^I_{adv}$ performs this query on a pseudo-identity PID_i, C_1 verifies whether the list L^{pk} comprises the tuple (PID_i, h_i, PK_i, θ_i, β_i) for PID_i. If it does, C_1 responds $\mathcal{A}^I_{adv}$ with the previously defined public key PK_i; if it does not, C_1 retrieves the corresponding elements (PID_i, A_i, h_i,) from the list L^{H_1}, chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly, computes $U_i = \beta_i P_{pub1}$ and $R_i = \theta_i P_{pub1}$, and then sets public key as $PK_i = (R_i, U_i)$. C_1 sends it to $\mathcal{A}^I_{adv}$ and adds (PID_i, h_i, PK_i, θ_i, β_i) to the list L^{pk}.

- *Generate private key*: If this query is performed on a pseudo-identity PID_i, C_1 checks whether $PID_i = PID_i^*$. If $PID_i = PID_i^*$, C_1 terminates and outputs "failure", if not, then C_1 executes as follows.
  - If the list L^{pk} comprises the concerned tuple (PID_i, h_i, PK_i, θ_i, β_i) for the PID_i, C_1 then sets the private key as $SK_i = (\beta_i, \theta_i)$ and responds $\mathcal{A}_{adv}^{I}$ with SK_i.
  - If the list L^{pk} does not comprise, C_1 first executes *generate partial-private key* and the *set public key* oracles queries for the PID_i, after this, C_1 performs the aforementioned process and responds $\mathcal{A}_{adv}^{I}$ with $SK_i = (\beta_i, \theta_i)$.
- *Replace public key*: Suppose $\mathcal{A}_{adv}^{I}$ performs this query with an input (PID_i, PK_i'), C_1 checks whether the list L^{pk} contains the tuple (PID_i, h_i, PK_i, θ_i, β_i).
  - If it does, C_1 sets $PK_i = PK_i'$ and responds it to $\mathcal{A}_{adv}^{I}$ and also update the list L^{pk} with $(PID_i, h_i, PK_i, \theta_i, \beta_i')$. Here it is assumed that C_1 can get an additional secret value β_i' relating to the public key PK_i' which is replacing from $\mathcal{A}_{adv}^{I}$.
  - If it does not, C_1 then performs *set public key* oracle query to generate (PID_i, h_i, PK_i, β_i). C_1 then sets $PK_i = PK_i'$ and inserts $(PID_i, h_i, PK_i, \theta_i, \beta_i')$ into the list L^{pk}.
- H_2 Oracle: When $\mathcal{A}_{adv}^{I}$ executes this query for a pseudo-identity PID_i on message M_i with an input tuple (M_i, PID_i, PK_i, j_i), C_1 first checks whether the list L^{H_2} comprises the tuple (M_i, PID_i, PK_i, j_i) for PID_i. If it does, C_1 responds $\mathcal{A}_{adv}^{1}$ with the previously defined H_2 hash value; otherwise C_1 picks a number $j_i \in \mathbb{Z}_q^*$ randomly and sets $j_i = H_2(M_i, PID_i, PK_i, P_{pub})$. C_1 sends j_i to $\mathcal{A}_{adv}^{I}$ and saves (M_i, PID_i, PK_i, j_i) in the list L^{H_2}.
- *CL-sign*: When $\mathcal{A}_{adv}^{I}$ performs this query with an input (M_i, PID_i), C_1 chooses two numbers $(\theta_i, j_i) \in \mathbb{Z}_q^*$ randomly from the lists L^{pk} and L^{H_2} respectively and computes $R_i = \theta_i P_{pub1} - (\frac{1}{j_i})U_i$. If the tuple containing j_i already present in the list L^{H_2} then C_1 halts and terminates. C_1 then chooses another two random numbers $(\theta_i, j_i) \in \mathbb{Z}_q^*$ and tries again. C_1 generates the signature as $\sigma_i = (\frac{1}{j_i\theta_i})P_{pub2}$ on message M_i for the pseudo-identity PID_i and adds (M_i, PID_i, θ_i, PK_i, j_i) into the list L^{H_2}. Finally, C_1 responds $\mathcal{A}_{adv}^{I}$ with the signature σ_i. If responses to *CL-sign* oracle queries are valid then certainly the output σ_i is also a valid signature on message M_i for the PID_i. The following verification of the generated signature satisfies Eq. (2).

$$\begin{aligned}
\hat{e}(\sigma_i, j_i R_i + U_i) &= \hat{e}\left(\left(\frac{1}{j_i\theta_i}\right)P_{pub2}, j_i\left(\theta_i P_{pub1} - \left(\frac{1}{j_i}\right)U_i\right) + U_i\right)\\
&= \hat{e}\left(\left(\frac{1}{j_i(s+h_i\gamma_i)}\right)P_{pub2}, j_i\left((s+h_i\gamma_i)P_{pub1} - \left(\frac{1}{j_i}\right)U_i\right) + U_i\right)\\
&= \hat{e}\left(\left(\frac{1}{j_i s + j_i h_i\gamma_i}\right)P_{pub2}, (j_i s + j_i h_i\gamma_i)P_{pub1}\right)\\
&= \hat{e}\left(\left(\frac{1}{(j_i s + j_i h_i\gamma_i)s}\right)P, (j_i s + j_i h_i\gamma_i)sP\right)\\
&= \hat{e}(P,P)^{\frac{1}{(j_i s+j_i h_i\gamma_i)s}(j_i s+j_i h_i\gamma_i)s}\\
&= \hat{e}(P,P)\\
&= g
\end{aligned}$$

Eventually, $\mathcal{A}_{adv}^{I}$ stops and computes a signature σ_i^* on a message M_i^* (notice no signature query was executed on message M_i^* before) for the pseudo-identity PID_i^* to satisfy the equation $e(\sigma_i^*, j_i^* R_i^* + U_i^*) = g$. If $PID_i \neq PID_i^*$, C_1 halts and outputs "failure"; otherwise, C_1 retrieves the tuples $(PID_i^*, h^*, PK_i^*, \beta_i^*)$ and $(M_i^*, PID_i^*, h^*, PK_i^*, j^*)$ from the lists L^{pk} and L^{H_2}, respectively and computes as follows.

$$\begin{aligned}
\hat{e}(\sigma_i^*, j^* R_i^* + U_i^*) &= \hat{e}\left(\sigma_i^*, j^*\theta_i^* P_{pub1} + \beta_i^* P_{pub1}\right)\\
&= \hat{e}\left(\sigma_i^*, (j^*\theta_i^* + \beta_i^*)P_{pub1}\right)\\
&= \hat{e}\left((j^*\theta_i^* + \beta_i^*)\sigma_i^*, P_{pub1}\right)\\
&= \hat{e}\left(j^*(s + h_i^*\gamma_i^*) + \beta_i^*)\sigma_i^*, P_{pub1}\right)\\
&= \hat{e}(P,P)\\
&= g
\end{aligned}$$

Thus C_1 can generate $(\frac{1}{j^*(s+h_i^*\gamma_i^*)+\beta_i^*})P_{pub1} = \sigma_i^*$ successfully and provides a solution as $(h^*, (\frac{1}{j^*(s+h_i^*\gamma_i^*)+\beta_i^*})P_{pub1})$, where $h^* \notin (h_1, h_2, \ldots, h_{q_{psk}})$. Therefore, C_1 can break k-CAA problem in G_1.

To illustrate the probability that C_1 breaks k-CAA problem in G_1 from [40]. The answers to H_1 and H_2 oracles queries for $\mathcal{A}_{adv}^{I}$ are valid. The answers to H_1 and H_2 oracles queries are impossible to differentiate from the real life. Each answer is consistently random and separately distributed in $\mathbb{Z}_q^*$. The events E_1, E_2 and E_3 do not happen if replies to *generate partial-private key, generate private key* and *CL-sign* oracles queries are all valid. In addition to this, when $\mathcal{A}_{adv}^{I}$ tries to extract a valid signature and event E_4 does not occur, then in this case, C_1 can provide solution to k-CAA problem. Therefore, C_1 can solve the k-CAA problem effectively, if all the events E_1, E_2, E_3, and E_4 do not happen. The following illustration provides the probability for the above events.

- E_1: For *generate partial-private key* oracle query, C_1 halts and outputs "failure", if $PID_i = PID_i^*$.
- E_2: For *generate private key* oracle query, C_1 halts and outputs "failure", if $PID_i = PID_i^*$.
- E_3: For *CL-sign* query, C_1 halts and outputs "failure", if $PID_i = PID_i^*$.
- E_4: $\mathcal{A}_{adv}^{I}$ generates a valid signature σ_i on M_i^* for PID_i^*, if $PID_i^* \neq PID_i$, C_1 halts and outputs "failure".

From the simulation, we have

$$Pr[E_1 \wedge E_2 \wedge E_3 \wedge E_4] = \left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

$\mathcal{A}_{adv}^{I}$ can generate a valid signature without executing H_2 queries is the only event which can happen. It is stated that the probability for $\mathcal{A}_{adv}^{I}$ to generate a valid signature without executing H_2 queries is at most $\frac{1}{2^k}$. From the aforementioned discussion, we concluded that C_1 can break the k-CAA problem in random oracle model with the following advantage.

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{psk}+q_{sk}+q_{sig}+1}$$

within the time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$. Therefore, in random oracle model, the proposed CL-PKS scheme is secure against Type-I adversary $\mathcal{A}_{adv}^{I}$ under the assumption that k- problem in G_1 cannot be tractable. □

Lemma 2. *If an adversary $\mathcal{A}_{adv}^{II}$ with non-negligible advantage ε against the EUF-CMA-II security of proposed CL-PKS scheme running in time t and performing q_{H_1} and q_{H_2} queries to H_1 and H_1 random oracles and q_{sk}, q_{pk}, and q_{sig} queries to the generate private key, set public key, and CL-sign oracles, then a challenger C_2 is constructed to solve the Inv-CDH problem with the following advantage.*

$$\varepsilon' \geq \left(\varepsilon - \frac{1}{2^k}\right)\left(\frac{q_{H_1} - 1}{q_{H_1}}\right)^{q_{sk}+q_{sig}+1}$$

within time $t' < t + (2q_{pk} + q_{sig})t_{sm} + q_{sig}t_{inv}$, where t_{sm} is the time needed for the computation of a scalar multiplication in G_1 and t_{inv} is the time needed for the computation of inversion in G_1.

Proof of Theorem. We consider an adversary $\mathcal{A}_{adv}^{II}$ of Type-II which breaks our proposed CL-PKS scheme. We consider a challenger C_2 to exploit $\mathcal{A}_{adv}^{II}$ for solving the *Inv*-CDH problem in G_1. $\mathcal{A}_{adv}^{II}$ and C_2 perform the adversary and challenger role respectively. C_2 accepts a challenge which contains random instance of *Inv*-CDH problem, i.e. given $P \in G_1$, $j^* \in \mathbb{Z}_q^*$ and $(j^* + \beta)P_{pub1}$, where β is unknown to C_2, the task of C_2 is to compute $(\frac{1}{j^*\theta_i + \beta})P_{pub1} \in G_1$. We consider two random oracles (i.e. H_1 and H_2) which are queried by the $\mathcal{A}_{adv}^{II}$. We also assume that a hash query for the pseudo-identity PID_i to H_1 oracle is performed before the keys extraction or signing queries for that pseudo-identity PID_i. C_2 keeps two hash lists L^{H_1} and L^{H_2} and a public key list L^{pk} which are initially empty. C_2 interacts with $\mathcal{A}_{adv}^{II}$, responds to all the queries of $\mathcal{A}_{adv}^{II}$, and

records these in the lists L^{H_1} and L^{H_2} and L^{pk} throughout Game II which are as follows.

- *Setup*: This algorithm is run by C_2 which inputs a security parameter 1^k for $k \in N$ and responds with system public parameters $params = (G_1, G_2, P, g, P_{pub}, H_1, H_2)$ and master secret key $s \in \mathbb{Z}_q^*$, where $g = \hat{e}(P, P)$ and $P_{pub1} = sP$, and sets $X = \beta P_{pub1}$, where $\beta \in \mathbb{Z}_q^*$. Note the secret key s is unknown to C_2. C_2 selects a pseudo-identity PID_i^* randomly as a challenged pseudo-identity for $\mathcal{A}_{adv}^{II}$ in this game and sends the *params* and s to $\mathcal{A}_{adv}^{II}$.
- H_1 Oracle: When $\mathcal{A}_{adv}^{II}$ with a pseudo-identity PID_i performs this query to H_1 oracle, C_2 checks whether the list L^{H_1} comprises (PID_i, A_i, h_i) for the PID_i. If it does, C_2 responds $\mathcal{A}_{adv}^{II}$ with a previously defined H_1 hash value; if it does not, C_2 select a number $h_i \in Z_q^*$ randomly and responds $\mathcal{A}_{adv}^{II}$ with h_i. C_2 then inserts (PID_i, A_i, h_i) into the list L^{H_1}.
- *Set public key*: Suppose $\mathcal{A}_{adv}^{II}$ performs this query on a pseudo-identity PID_i, C_2 verifies whether the list L^{pk} comprises the tuple (PID_i, h_i, PK_i, θ_i, β_i) for PID_i. If it does, C_2 responds $\mathcal{A}_{adv}^{II}$ with the previously defined public key PK_i; if it does not, C_2 first retrieves the corresponding elements (PID_i, A_i, h_i) from the list L^{H_1}, chooses a number $\beta_i \in \mathbb{Z}_q^*$ randomly, computes $X_i = \beta_i P_{pub1}$ and $R_i = \theta_i P_{pub1}$, and then sets public as $PK_i = (R_i, X_i)$. C_2 sends it to $\mathcal{A}_{adv}^{II}$ and adds (PID_i, h_i, PK_i, θ_i, β_i) to the list L^{pk}.
- *Generate private key*: If this query is performed on a pseudo-identity PID_i, C_2 checks whether $PID_i = PID_i^*$. If $PID_i = PID_i^*$, C_2 terminates and outputs "failure", if not, then C_2 executes as follows.
  - If the list L^{pk} comprises the concerned tuple (PID_i, h_i, PK_i, θ_i, β_i) for PID_i, C_2 then sets the private key as $SK_i = (\beta_i, \theta_i)$ and responds $\mathcal{A}_{adv}^{II}$ with SK_i.
  - If the list L^{pk} does not comprise, C_2 first executes *generate partial-private key* and *set public key* oracles queries for the PID_i, after this, C_2 performs the aforementioned process and responds $\mathcal{A}_{adv}^{II}$ with SK_i.
- H_2 Oracle: When $\mathcal{A}_{adv}^{II}$ executes this query for a pseudo-identity PID_i on message M_i with an input tuple (M_i, PID_i, PK_i, j_i), C_2 first checks whether the list L^{H_2} comprises the tuple (M_i, PID_i, PK_i, j_i) for PID_i. If it does, C_2 responds $\mathcal{A}_{adv}^{II}$ with the previously defined H_2 hash value; otherwise C_2 picks a number $j_i \in Z_q^*$ randomly and sets $j_i = H_2(M_i, PID_i, PK_i, P_{pub})$. C_2 sends j_i to $\mathcal{A}_{adv}^{II}$ and saves (M_i, PID_i, PK_i, j_i) in the list L^{H_2}.
- *CL-sign*: When $\mathcal{A}_{adv}^{II}$ performs this query with an input (M_i, PID_i), C_2 chooses two numbers $(\theta_i, j_i) \in \mathbb{Z}_q^*$ randomly from the lists L^{pk} and L^{H_2} respectively and computes $R_i = \theta_i P_{pub1} - (\frac{1}{j_i})U_i$. If the tuple containing j_i already present in the list L^{H_2} then C_2 halts and terminates and chooses another random numbers $(\theta_i, j_i) \in Z_q^*$ and tries again. C_2 generates a signature as $\sigma_i = (\frac{1}{j_i \theta_i})P_{pub2}$ on message M_i for the pseudo-identity PID_i and adds (M_i, PID_i, θ_i, PK_i, j_i) into the list L^{H_2}. Finally, C_2 responds $\mathcal{A}_{adv}^{II}$ with the signature σ_i. If responses to *CL-sign* oracle queries are valid then output σ_i is also a valid signature on message M_i for the pseudo-identity PID_i. The verification of this signature σ_i is similar to verification of signature in Lemma 1 which satisfies Eq. (2).

Eventually, $\mathcal{A}_{adv}^{II}$ stops and computes a signature σ_i^* on a message M_i^* (notice no signature query was generated on message M_i^* before) for the pseudo-identity PID_i^* with respect to public key PK_i^* to satisfy the equation $e(\sigma_i^*, j_i^* R_i^* + U_i^*) = g$. If $PID_i^* \neq PID_i^*$, C_2 halts and outputs "failure"; otherwise, C_2 retrieves the tuples ($PID_i^*, h^*, PK_i^*, \beta_i$) and ($M_i^*, PID_i^*, h^*, PK_i, j^*$) from the lists L^{pk} and L^{H_2}, respectively. Note, C_2 has already computed $X = \beta P_{pub1}$ and sets $PK_i^* = (R_i, X)$ and therefore computes as follows.

$$\begin{aligned}\hat{e}(\sigma_i^*, j^* R_i + U_i) &= \hat{e}\big(\sigma_i^*, j^* R_i + X\big)\\ &= \hat{e}\big(\sigma_i^*, (j^*\theta_i + \beta)P_{pub1}\big)\\ &= \hat{e}\big((j^*\theta_i + \beta)\sigma_i^*, P_{pub1}\big)\\ &= \hat{e}(P, P)\\ &= g\end{aligned}$$

Thus C_1 can generate $(\frac{1}{j^*\theta_i + \beta})P_{pub1} = \sigma_i^*$ successfully and provides a solution as $(\frac{1}{j^*\theta_i + \beta})P_{pub1}$. Therefore, C_1 can break *Inv*-CDH problem in G_1. The discussion of the probability of C_2 advantage and the required running time is analogous to that of the Lemma 1. Therefore, in the random oracle model, the proposed CL-PKS scheme is EUF-CMA secure against Type-II adversary with the assumption that *Inv*-CDH problem in G_1 is intractable. □

6.2. Security requirements fulfilled by proposed CL-PKS scheme

From the aforementioned discussion, we are going to extract and show the security requirements for our proposed CL-PKS scheme.

1. **Source authentication and message integrity:** The RSU has the capability to authenticate traffic-related messages from the vehicle V_i that they were certainly sent by legitimate vehicle for the concerned RSU without modification. The proposed CL-PKS scheme uses pseudo-identity PID_i which is extracted from sender real identity *RID*. Therefore, our scheme supports pseudo-identity PID_i authentication which ensures source authentication. From the Definition 1, we also know that no polynomial time adversary can make a duplicate authentic message due to the hardness of DL problem. Thus, the RSU can validates the integrity of message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) if the verification of this equation $e(\sigma_i, j_i R_i + U_i) = g$ holds. Therefore, our CL-PKS scheme provides message integrity in VANETs.
2. **Identity privacy preservation:** Each pseudo-identity PID_i in our proposed CL-PKS scheme is composed TRA's master secret key x and the secret key α_i chosen by vehicle V_i. Therefore, the values of these secret keys x and α_i are only known by TA and vehicle V_i, respectively. The vehicle V_i real identity *RID* is also unknown to KGC to preserve privacy. From Definition 2, it is impossible for an adversary to generate $xPID_{i,1}$ and extract the real identity *RID* of the sender vehicle without knowing x or α_i due to CDH problem. For instance, $T_{pub} = xP$, $PID_{i,1} = \alpha_i P$, $PID_{i,2} = RID \oplus H_0(xPID_{i,1}, T_{pub}, T_i)$ and $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$. To extract the real identity *RID* of the vehicle V_i from $PID_{i,2} = RID \oplus H_1(xPID_{i,1}, T_{pub})$, the malicious adversary has to compute $xPID_{i,1} = x\alpha_i P$ from $T_{pub} = xP$ and $PID_{i,1} = \alpha_i P$. It means that the adversary cannot extract real identity *RID* from the mentioned computation due to the hardness of CDH problem. Therefore, according to Definition 2, we decide that our proposed CL-PKS scheme provides identity privacy preservation in V2I.
3. **Traceability and revocability:** The proposed CL-PKS scheme provides identity privacy preservation conditionally in VANETs. The TRA traces and reveals real identity of malicious vehicle from its database. For instance, the TRA extracts the real identity *RID* from a vehicle V_i pseudo-identity PID_i when the traffic-related message or signature found disputed. In the proposed CL-PKS scheme, the master secret key x of TRA is used to extract the real identity *RID* of vehicle V_i from $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$ by computing the following.

$$\begin{aligned}RID &= PID_{i,2} \oplus H_0(xPID_{i,1}, T_{pub})\\ &= RID \oplus H_0(xPID_{i,1}, T_{pub}) \oplus H_0(xPID_{i,1}, T_{pub})\\ &= RID\end{aligned}$$

Thus the TRA can recover the real identity *RID* of vehicle V_i, if the traffic-related message or signature is found disputed. Therefore, the proposed CL-PKS scheme preserves privacy conditionally it means that our scheme provides traceability.

After this, the TRA revokes such pseudo-identity PID_i and stores it in RPID-BC. For this purpose, every entity in VANETs can easily

check the RPID-BC before dealing request of malicious node whose pseudo-identity PID_i has been revoked. For instance, an RSU receives a CL-Signature σ_i from a malicious vehicle V_i whose pseudo-identity PID_i has been revoked. After checking the time-stamp in PID_i, the RSU looks into RPID-BC to find pseudo-identity PID_i. If such pseudo-identity PID_i is found present then RSU rejects the concerned CL-Signature σ_i. Similarly the KGC can also check the RPID-BC to find such revoked pseudo-identity PID_i. Therefore, RPID-BC provides revocation transparency efficiently with the PoP and PoA in our proposed CL-PKS scheme.

4. **Un-linkability**: In vehicular communication, un-linkability is defined as, no malicious vehicles or RSUs or third parties are able to link messages that they were sent from the same source (i.e. vehicle) or also do not link messages to a specific vehicle, license plate number, other personal information about driver identity, etc. To get un-linkability, the proposed CL-PKS scheme utilizes pseudo-identity PID_i. For instance, a malicious adversary takes interest in knowing that a specific vehicle V_i have generated two messages M and M'. The adversary cannot do this successfully because it will have to pass from the CDH problem in Definition 2. The message M and message M' are signed by different private keys SK_i. The partial private key PSK_i is based on PID_i and there is no any link between the PID_i. The construction of private key SK_i and also the over all CL-Signature σ_i is based on the selection of secure random numbers. For instance, the vehicle V_i generates $\alpha_i \in \mathbb{Z}_q^*$, where as $PID_{i,1} = \alpha_i P$, $PID_{i,2} = RID \oplus H_0(xPID_{i,1}, T_{pub}, T_i)$ and $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$, and KGC generates $\gamma_i \in \mathbb{Z}_q^*$, where as $A_i = \gamma_i P$ and $\theta_i = (s + h_i\gamma_i) \bmod q$ and also vehicle generates its secret key β_i and its corresponding public key PK_i. Due to these random numbers α_i, γ_i and β_i, no malicious adversary can be able to link pseudo-identities or messages to ensure that these belongs to the specific vehicle V_i. Hence, the proposed CL-PKS scheme provides un-linkability in VANETs.
5. **Authority distribution**: The IDC faces key escrowing problem incurred from PKG because it can use private key of the user. To overcome this, Shamir introduced CL-PKC [19]. In CLC, power centralization for private key generation is partially distributed from KGC to the user. The KGC extracts partial private key PSK_i and sends it to the vehicle V_i. Vehicle V_i then generates its private key SK_i and uses both PSK_i and SK_i for signing the message M_i. It means that vehicle V_i have the authority to generate the full private key. Hence, KGC cannot know and extract the full private of the vehicle V_i and therefore removed the inherent problem of key escrowing from ID-PKC.
6. **Resistance against attacks**: Our proposed CL-PKS scheme resists against several different attack types (i.e. modification attack, impersonation attack, replay attack, and man-in-the-middle attack). These are given as follows.
   - *Impersonation attack*: According to the Theorem 1, an adversary cannot create a message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) on behalf of the vehicle V_i to RSU because the message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) is authenticated by the RSU and can identify the impersonation attack easily by checking whether the equation $\hat{e}(\sigma_i, j_i R_i + U_i) = g$ holds or not. Hence, our proposed CL-PKS scheme has the ability to resist the impersonation attack in VANETs.
   - *Modification attack*: The CL-Signature $\sigma_i = (\frac{1}{j_i\theta_i + \beta_i})P_{pub2}$ on the message M_i for the PID_i and the corresponding public key PK_i is generated from vehicle V_i. According to Theorem 1, any modification in message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) can be detected by checking the equation $\hat{e}(\sigma_i, j_i R_i + U_i) = g$. Therefore, the proposed CL-PKS scheme can protect V2I communication from modification attack.
   - *Man-In-The-Middle attack*: As the proposed scheme is mainly based on authentication and from the analysis it is ensured that source and message authentication is carried out between sender vehicle V_i and the receiver RSU. Hence, our proposed CL-PKS scheme has the ability to resist man-in-the-middle attack in V2I communication.
   - *Replay attack*: The time-stamps T_i and t_i are inserted in $PID_i = (PID_{i,1}, PID_{i,2}, T_i)$ and in message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i), respectively which can allow the RSU to identify a replay attack by checking the freshness of time-stamps T_i and t_i. Hence, our proposed CL-PKS scheme has the ability to resist replay attack in V2I communication.

7. Performance analysis

The analysis and comparison of performance efficiency of our proposed CL-PKS scheme with respect to computation and communication costs for V2I communication are given in this section.

7.1. Computational cost analysis

We analyse our CL-PKS scheme in terms of computational cost and then compare it with the recent relevant CL-Signatures schemes [28–31] with respect to signature generation and its verification for V2I communication in this section. The proposed CL-PKS scheme has been constructed with bilinear pairing while the other schemes have also been constructed with bilinear pairings. To analyze the proposed CL-PKS scheme and the related CL-Signature schemes [28–31] computation costs, we denote the T_{bp} to represents the execution time required for a bilinear pairing operation, T_{sm} the execution time required for a scalar multiplication operation in G_1, T_{pa} the execution time required for a point addition operation in G_1, and T_{mtp} the execution time required for a map-to-point hash operation in G_1. Note in the proposed CL-PKS and in other schemes, we do not consider the execution time required for a one-way general hash function operation in $\mathbb{Z}_q^*$ because it takes a negligible amount of time in a signature generation and its verification process.

To compute the execution time of the aforementioned cryptographic operations, we adopt the experiment that has been performed in [20] using the MIRACL library [47] on hardware platform comprises an Intel I7-4770 3.40 GHz machine, 4 gigabytes memory, and operates Windows 7 operating system. MIRACL library is a famous cryptographic library which is used extensively to efficiently perform mathematical operations underlying pairing based cryptography in some environments. We utilize bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$, sets 80 bits security level [46] and performs type A pairing on the super-singular elliptic curve E. The equation $y^2 \equiv (x^3 + x) \bmod p$ defines E, where $p \equiv 3 \bmod 4$, the embedding degree is 2, and the order of G_1 is q. The p and q are 512 bits prime number and 160 bits Solinas prime number respectively while the equation $p + 1 = 12qr$ holds. Table 2 provides the execution times for the aforementioned mathematical operations. The detailed comparison for computational costs is shown in Table 3.

From Table 3, we can see that a vehicle V_i in Malhi et al. [28] generated a signature which comprises four scalar multiplication and two point addition operations while for the verification of this signature, an RSU requires three bilinear pairing, three scalar multiplication, and one point addition operations. According to Table 2, the vehicle V_i in Malhi et al.'s scheme [28] requires $4T_{sm} + 2T_{pa} \approx 6.8502$ ms for message signing while for corresponding signature verifying, the RSU needs $3T_{bp} + 3T_{sm} + 1T_{pa} \approx 17.7671$ ms. Therefore, in message signing and signature verifying, the total computation cost is approximately 24.6173 ms. Similarly, the RSU needs to execute $3T_{bp}$ bilinear pairing, $3nT_{sm}$ scalar multiplication, and nT_{pa} point addition operations for n signature verification

Table 2
Cryptographic operations execution times.

Cryptographic Operation:	T_{bp}	T_{sm}	T_{pa}	T_{mtp}
Execution Time (ms):	4.211	1.709	0.0071	4.406

Table 3
Computation costs comparison.

Ref.	Message-Signing	Single-Sig Verify	n-Sig Verify	Verification Function	Security
Malhi et al. [28]	$4T_{sm} + 2T_{pa} \approx 6.8502$ ms	$3T_{bp} + 3T_{sm} + 1T_{pa} \approx 17.7671$ ms	$3T_{bp} + 3nT_{sm} + nT_{pa} \approx 5.1341n + 12.633$ ms	Aggregate	√
Horng et al. [29]	$2T_{sm} + 1T_{pa} \approx 3.4251$ ms	$3T_{bp} + 1T_{sm} + 1T_{pa} + 1T_{mtp} \approx 18.7551$ ms	$3T_{bp} + nT_{sm} + nT_{pa} + nT_{mtp} \approx 6.1221n + 12.633$ ms	Batch and Aggregate	✗
Li et al. [30]	$2T_{sm} + 1T_{pa} + 1T_{mtp} \approx 7.8311$ ms	$3T_{bp} + 1T_{sm} + 1T_{pa} + 2T_{mtp} \approx 23.1611$ ms	$3T_{bp} + nT_{sm} + nT_{pa} + (n+1)T_{mtp} \approx 6.1221n + 17.039$ ms	Aggregate	√
Kumar et al. [31]	$4T_{sm} + 2T_{pa} + 1T_{mtp} \approx 11.2562$ ms	$4T_{bp} + 3T_{sm} + 1T_{mtp} \approx 26.377$ ms	$4T_{bp} + 3nT_{sm} + (n+1)T_{mtp} \approx 9.533n + 21.25$ ms	Aggregate	√
Proposed	$1T_{sm} \approx 1.709$ ms	$1T_{bp} + 1T_{sm} + 1T_{pa} \approx 5.9271$ ms	$1T_{bp} + nT_{sm} + nT_{pa} \approx 1.7161n + 4.211$ ms	Batch and Aggregate	√

on n message, where $n = 1, 2, 3, \ldots$ .. Therefore, the RSU in Malhi et al.'s scheme [28] needs $3T_{bp} + 3nT_{sm} + nT_{pa} \approx 5.1341n + 12.633$ ms to verify n signature on n message.

In Horng et al.'s scheme [29], the vehicle V_i requires $2T_{sm} + 1T_{pa} \approx 3.4251$ ms in message signing while for verifying the corresponding signature, the RSU needs $3T_{bp} + 1T_{sm} + 1T_{pa} + 1T_{mtp} \approx 18.7551$ ms. Therefore, the total computational cost for message signing and verifying the corresponding signature is approximately 22.1802 ms. Similarly, the RSU in Horng et al.'s scheme [29] needs $3T_{bp} + nT_{sm} + nT_{pa} + nT_{mtp} \approx 6.1221n + 12.633$ ms to verify n signature on n message.

The vehicle V_i in Li et al.'s scheme [30] requires $2T_{sm} + 1T_{pa} + 1T_{mtp} \approx 7.8311$ ms for message signing while for verifying the corresponding signature, the RSU needs $3T_{bp} + 1T_{sm} + 1T_{pa} + 2T_{mtp} \approx 23.1611$ ms. Therefore, the total computational cost in message signature and verifying of the corresponding signature is approximately 30.9922 ms. Similarly, the RSU in Li et al.'s scheme [30] needs $3T_{bp} + nT_{sm} + nT_{pa} + (n+1)T_{mtp} \approx 6.1221n + 17.039$ ms to verify n signature on n message.

In Kumar et al.'s scheme [31], the vehicle V_i requires $4T_{sm} + 2T_{pa} + 1T_{mtp} \approx 11.2562$ ms for message signing while for verifying the corresponding signature, the RSU needs $4T_{bp} + 3T_{sm} + 1T_{mtp} \approx 26.377$ ms. Therefore, the total computational cost for message signing and verifying the corresponding signature is approximately 37.6332 ms. Similarly, the RSU in Kumar et al.'s scheme [31] needs $4T_{bp} + 3nT_{sm} + (n+1)T_{mtp} \approx 9.533n + 21.25$ ms to verify n signature on n message.

Since under our proposed CL-PKS scheme, the vehicle V_i generates the signature on message with one scalar multiplication operation while the verification of the corresponding signature on message comprises one bilinear pairing, one scalar multiplication, and one point addition operations. According to Table 2, the vehicle V_i in CL-PKS scheme requires $1T_{sm} \approx 1.709$ ms to sign the message while to verify the corresponding signature on the message, the RSU needs $1T_{bp} + 1T_{sm} + 1T_{pa} \approx 5.9271$ ms. Thus, the total computational cost of CL-PKS scheme in message signing and in verifying the corresponding signature on the message is approximately 7.6361 ms. Similarly, the RSU in the proposed CL-PKS scheme needs $1T_{bp} + nT_{sm} + nT_{pa} \approx 1.7161n + 4.211$ ms to verify n signature on n message. The comparison of computational costs with respect to single message signing and single signature verifying are represented graphically in Figs. 4 and 5 respectively. The comparison of computational costs with respect to batch message signing and batch signature verifying are represented graphically in Figs. 6 and 7 respectively.

The improvement in performance efficiency of our proposed CL-PKS scheme in percentage with respect to message signing and signature verifying costs over Malhi et al.'s [28] scheme is about $\frac{6.8502-1.709}{6.8502} \times 100 \approx 75.05\%$ and $\frac{17.7671-5.9271}{17.7671} \times 100 \approx 66.64\%$, respectively. Similarly the improvement of CL-PKS scheme over Horng et al.'s scheme [29] in message signing and signature verifying on the message is about 50.10% and 68.40%, respectively. Improvement over Li et al.'s scheme [30] in message signing and verifying signature is about 78.18% and 74.41%, respectively. And improvement over Kumar et al.'s scheme [31] in message signing and signature verifying is about 84.82% and 77.53%, respectively. The comparison in percentage is shown in Table 4.

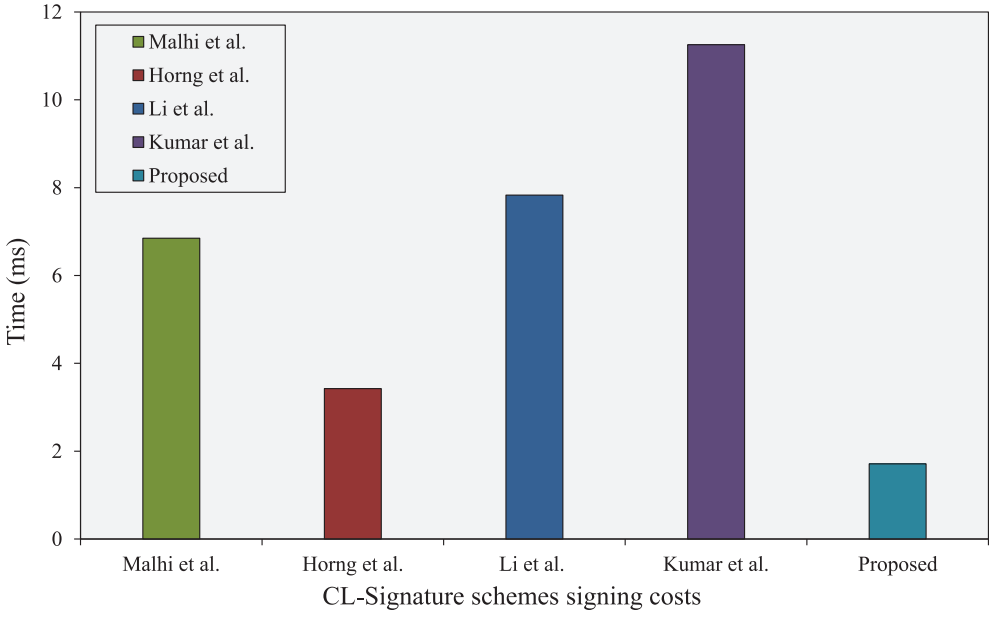

Fig. 4. Graphical representation of single message signing cost.

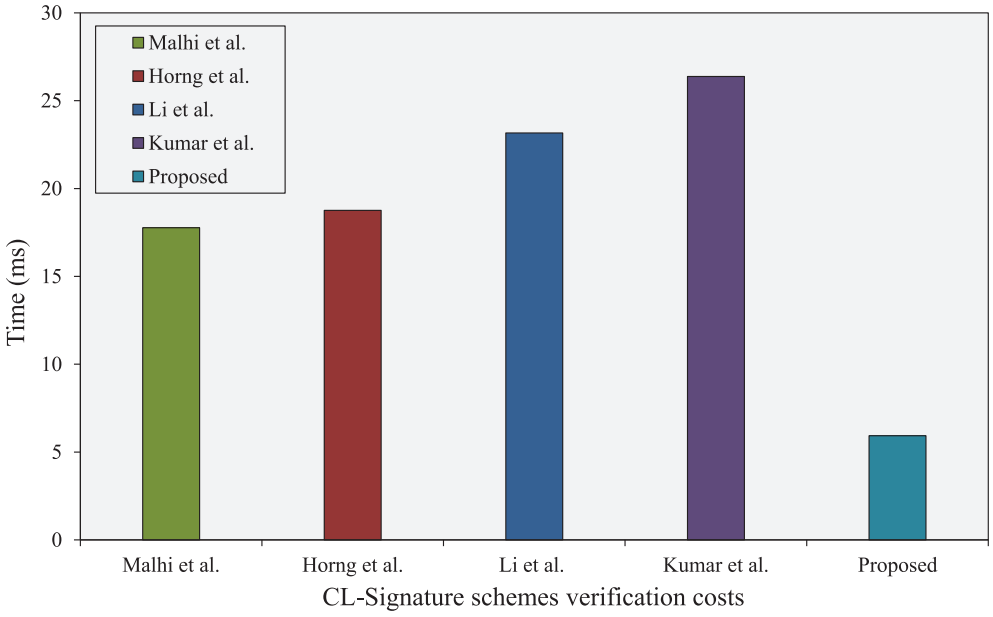

Fig. 5. Graphical representation of a single signature verification cost.

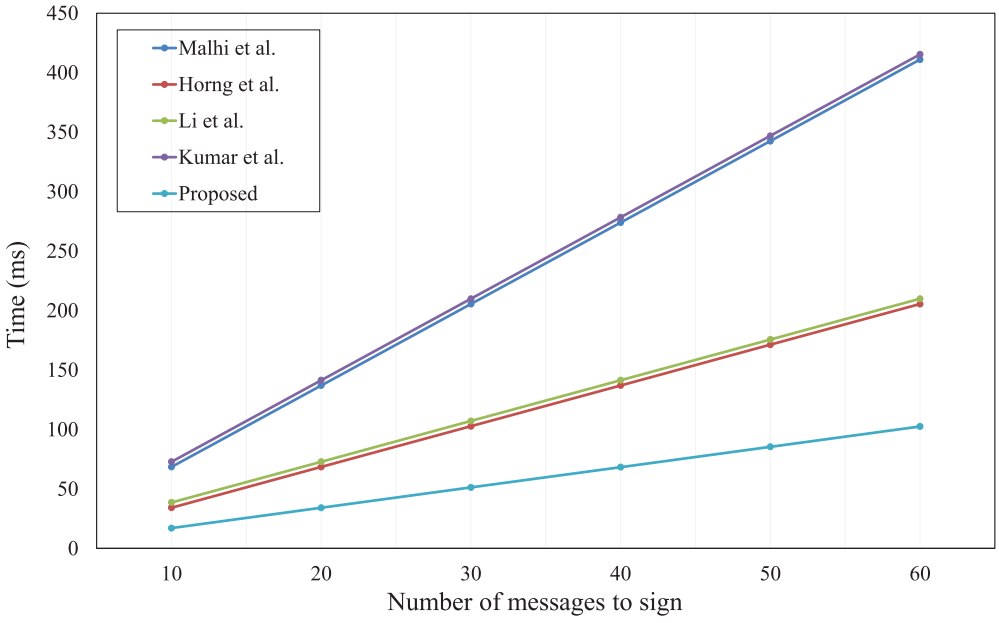

Fig. 6. Execution time for batch message signing.

Table 4
Improvement of the proposed CL-PKS scheme over the existing schemes in CL-Signature generation and verification.

Ref.	Msg-Sign	Single-Sig Verify	*n*-Sig Verify
Malhi et al. [28]	75.05%	66.64%	66.58%
Horng et al. [29]	50.10%	68.40%	71.57%
Li et al. [30]	78.18%	74.41%	72.29%
Kumar et al. [31]	84.82%	77.53%	81.85%

From Figs. 6 and 7 and also from Table 4, we observe that the traffic-related message signing delay and signature verifying delay of the proposed CL-PKS scheme are less than the existing CL-Signature schemes [28–31]. Therefore, our proposed CL-PKS scheme is more efficient with respect to reduced computational cost among the existing CL-Signature schemes [28–31] for VANETs. In addition to this, it is more suitable for bandwidth limited networks such as VANETs.

7.2. Communication cost analysis

To analyse the proposed CL-PKS scheme and related CL-Signature schemes [28–31] communication costs for VANETs, we consider that the status of the traffic-related message M_i in our CL-PKS scheme and the existing CL-Signature schemes are the same. Therefore, we analyse the communication cost by considering the size of the parameters such as pseudo-identity, public key, current time-stamp and CL-Signature of the vehicle. On 80-bit security level, the p size based on the equation $E : y^2 \equiv (x^3 + x) \bmod p$ will become 512 bits (64 bytes) and the size of G_1 elements will become $64 \times 2 = 128$ bytes (1024 bits). Furthermore, we consider 4 bytes and 20 bytes the size for the time-stamp and the size for general hash function or number in $\mathbb{Z}_q^*$, respectively.

The vehicle V_i in Malhi et al.'s [28] scheme broadcasts pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i \in G_1$, time-stamp T_i, and CL-Signature $\sigma_i = (S_i, V_i) \in G_1$ to RSU in VANETs. There-

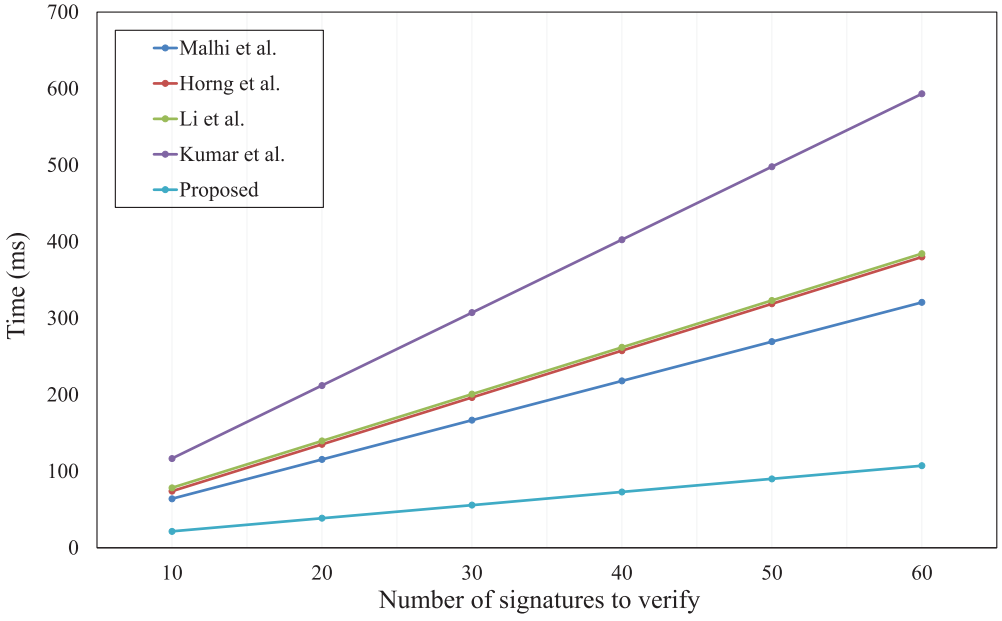

Fig. 7. Execution time for batch signature verification.

Table 5
Communication costs comparison.

Ref.	Single-Msg. Transmission	*n*-Msg. Transmission
Malhi et al. [28]	536 bytes	536*n* bytes
Horng et al. [29]	536 bytes	536*n* bytes
Li et al. [30]	536 bytes	536*n* bytes
Kumar et al. [31]	536 bytes	536*n* bytes
CL-PKS scheme	536 bytes	536*n* bytes

fore, the communication cost for the vehicle V_i in Malhi et al.'s [28] scheme is $128 \times 4 + 20 + 4 = 536$ bytes. Similarly, in the schemes of Horng et al. [29] and Li et al. [30], the vehicle V_i broadcasts pseudo-identity $PID_i = (PID_{i,1}, PID_{i,2}) \in G_1$, public key $PK_i \in G_1$, time-stamp T_i, and the CL-Signature $\sigma_i = (S_i, V_i) \in G_1$ to RSU. Therefore, the communication costs for the vehicle V_i in Horng et al.'s [29] and Li et al.'s [30] schemes are $128 \times 4 + 20 + 4 = 536$ bytes. The vehicle V_i in Kumar et al. [31] scheme broadcasts pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i \in G_1$, time-stamp T_i, and CL-Signature $\sigma_i = (S_i, V_i) \in G_1$ to RSU in VANETs. Therefore, the communication cost for the vehicle V_i in Kumar et al.'s [31] scheme is $128 \times 4 + 20 + 4 = 536$ bytes.

In our proposed CL-PKS scheme, the vehicle V_i broadcasts pseudo-identity $PID_i = (PID_{i,1} \in G_1, PID_{i,2} \in \mathbb{Z}_q^*)$, public key $PK_i = (R_i, U_i) \in G_1$, time-stamp T_i, and signature $\sigma_i = (\frac{1}{j_i\theta_i + \beta_i})P_{pub2} \in G_1$ to the RSU in VANETs. Therefore, the communication cost for the vehicle V_i in the proposed CL-PKS scheme is $128 \times 4 + 20 + 4 = 536$ bytes. The proposed CL-PKS scheme has the same communication cost as compared to the CL-Signature schemes [28–31]. Therefore, with the reduction in computation cost by the proposed CL-PKS scheme, the communication cost is also maintained. Hence, our proposed CL-PKS scheme is suitable for bandwidth limited infrastructure especially for VANETs and can perform efficiently in V2I communication. The comparison of communication costs is provided in Table 5.

8. Application of proposed CL-PKS scheme in VANETs

In this section, the application of our proposed scheme in VANETs is given. VANETs have been emerging and help to improve traffic on roads by broadcasting traffic-related information between vehicles and infrastructure. Fig. 8 depicts a secure V2I communication model for VANETs. The proposed model is composed of network nodes such as TRA, KGC, RSU, and vehicle with OBU. In addition to this, we also included two blockchains (i.e. PID-BC and RPID-BC) in the proposed model. TRA and KGC first execute the *Setup* algorithm to initialize the system public parameters. Then TRA executes the *PIDGen* algorithm to generate pseudo-identity PID_i upon the request of the vehicle V_i. Note, the vehicle V_i presents real identity *RID* and some other information which he obtained from the Motor Vehicle Department (MVD) to TRA. After verifying the *RID* uniqueness, TRA computes PID_i and sends it to KGC and vehicle V_i via the secure channel. TRA also signs PID_i using its master secret key x, inserts it in the database for future reference and also adds it in the PID-BC.

Then KGC executes the *PPKGen* to generate a partial private key PSK_i for the vehicle V_i upon the request of TRA. KGC looks up the PID-BC and RPID-BC using T_{pub} to ensure that PID_i is present in PID-BC and absent in RPID-BC, which means that PID_i has been assigned to vehcile V_i and has not been revoked by TRA. CMT and LMT in PID-BC and RPID-BC present the PoP and PoA of PID_i for the KGC with $O(\log^N)$ efficiency. The KGC then computes the partial private key PSK_i and sends it to vehicle V_i.

After this, vehicle V_i checks the authenticity of the partial private key PSK_i and executes *SPKGen* algorithm to generate a private key SK_i and a corresponding public key PK_i.

The vehicle V_i then executes the *CLSigGen* algorithm to generate a CL-Signature $\sigma_i = (\frac{1}{j_i\theta_i + \beta_i})P_{pub2}$ on a message M_i and sends the message-signature tuple (M_i, PID_i, PK_i, σ_i, t_i) to the nearest RSU. According to the DSRC protocol [36,37], the steps above are repeated every 100–300 ms within the VANETs.

The verifier RSU receives this and verifies the CL-Signature σ_i on the message M_i to guarantee that the corresponding vehicle V_i is not trying to pretend to be any other legitimate vehicle or broadcast bogus messages. RSU checks the time-stamps and then look up the PID-BC and RPID-BC for the pseudo-identity PID_i using master public key T_{pub} of TRA to ensure that the received PID_i is present in PID-BC and absent in RPID-BC. After this, RSU executes *CLSigVerify* algorithm to verify the authenticity and integrity of traffic-related message M_i generated by vehicle V_i by single signature verification and batch signature verification functions. In batch signature verification function, signatures on multiple traffic-related messages generated from different vehicles are verified by a single RSU, simultaneously. This function reduces pairing operations. The computational cost at the RSU is

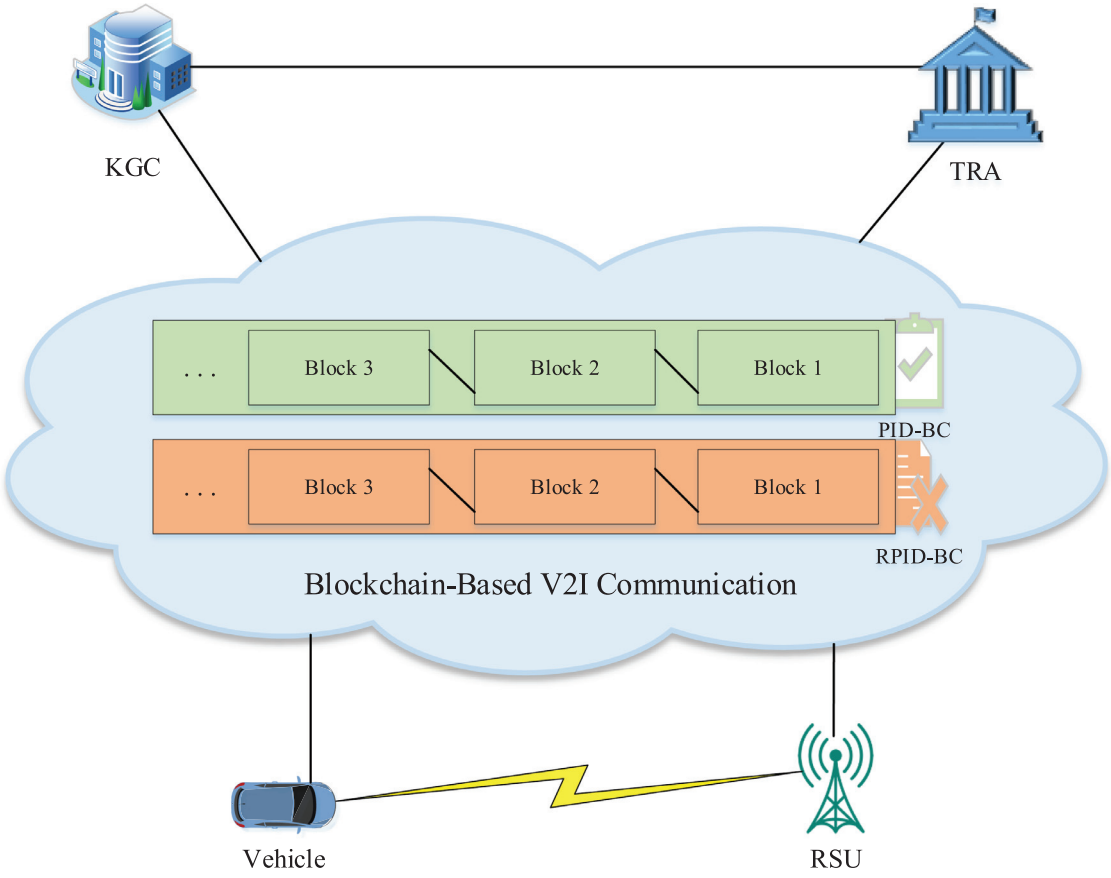

Fig. 8. Secure and efficient V2I communication model.

reduced due to only one pairing operation in the signature verification which makes efficient the overall performance of V2I communication in VANETs. The blockchain in the proposed model allows each network node to check the source of traffic-related message that either it is revoked or not. This act efficiently provides the revocation transparency before verifying the signature on the traffic-related message in V2I communication.

We also considered that RSU is connected to the Application Server to act as an aggregate signature generator, once received multiple message-signature tuples (M_1, PID_1, PK_1, σ_1, t_1), (M_2, PID_2, PK_2, σ_2, t_2),...., (M_n, PID_n, PK_n, σ_n, t_n) generated from n vehicle ($V_1, V_2, \ldots ., V_n$). RSU executes *ACLSigGen* algorithm to aggregate a bundle of CL-Signatures into single one and sends it to Application Server for further verification.

Upon receiving the CL aggregate signature $\sigma_{agg} = (\sigma_1, \sigma_2, \ldots ., \sigma_n)$, the Application Server executes *ACLSigVerify* algorithm to verify the CL aggregate signature σ_{agg}. The Application Server then distributes the verified traffic-related information to RSUs. Then each RSU broadcasts such information in its coverage in VANETs.

The aggregate signature generation is useful in those areas where signatures on traffic-related messages generated from different sources need to be compressed. This characteristic is very suitable for authentication in a resource-limited infrastructure [17]. In our proposed model, it significantly decreases the computational overhead in CL-Signatures verification and the storage overhead in transmitting CL-Signatures. The traffic-related information generated from the Application Server is more authentic for the rest of the nodes in VANETs. Hence, the model based on the proposed CL-PKS scheme is suitable for bandwidth limited infrastructure especially for VANETs and can perform efficiently in V2I communication.

9. Conclusions and future work

In this paper, we proposed an efficient CL-PKS scheme which uses bilinear pairing and provides conditional privacy-preserving authentication for V2I communication in VANETs. This scheme does not use map-to-point hash functions and reduces the computational cost produced from signatures' verification on messages at the RSU and makes efficient performance of V2I communication. This is because only one bilinear pairing operation exist in the verification of a signature. The proposed CL-PKS scheme supports both batch signature verification and aggregate signature verification functions to enhance more the performance of V2I communication in VANETs. In addition to this, we included the blockchain in CL-PKS scheme to show the revocation transparency efficiently before verifying the signature on traffic-related message. Our proposed CL-PKS scheme provides EUF-CMA security against Type-I adversary and Type-II adversary in the random oracle model and is efficient with respect to computational cost in signature generation and verification in comparison with the existing CL-Signature schemes. Therefore, the proposed CL-PKS scheme is suitable for VANETs environment and can perform efficiently in V2I communication.

The future extension of this work is based on the design of CL-Signature scheme for V2V communication without bilinear pairing to support batch verification in a more efficient way in VANETs.

Declaration of Competing Interest

The authors declared that they have no conflicts of interest to this work.

Acknowledgments

The authors thank the editors and the anonymous reviewers for their valuable and enriching comments and suggestions. This work is supported by the National Natural Science Foundation of China under grant no. 61872058 and the Fundamental Research Funds for the Central Universities under grant no. ZYGX2016J081.

References

[1] IEEE, Trial-use standard for wireless access in vehicular environments (WAVE) - networking services, IEEE Std 1609.3–2007 (2007) 1–99.
[2] I. Ali, M. Faisal, S. Abbas, A survey on lightweight authentication schemes in vertical handoff, Int. J. Coop. Inf. Syst. 26 (1) (2017) 1630001.
[3] C. Zhang, Y. Zhang, Y. Fang, A coverage inference protocol for wireless sensor networks, IEEE Trans. Mob. Comput. 9 (6) (2010) 850–864.
[4] C. Zhang, X. Zhu, Y. Song, Y. Fang, A formal study of trust-based routing in wireless ad hoc networks, in: Proc. IEEE INFOCOM, 2010, pp. 1–9. San Diego, CA.
[5] J.A. Misener, Vehicle-infrastructure integration (VII) and satety: rubber and radio meets the road in California, UC Berkeley Transp. Lib. Intellimotion 11 (2) (2005) 1–3.
[6] S. Zeadally, R. Hunt, Y.S. Chen, A. Irwin, A. Hassan, Vehicular ad hoc networks (VANETs): status, results, and challenges, Telecommun. Syst. 50 (4) (2012) 217–241.
[7] F. Qu, Z. Wu, F.Y. Wang, W. Cho, A security and privacy review of VANETs, IEEE Trans. Intell. Transp. Syst. 16 (6) (2015) 2985–2996.
[8] S.S. Manvi, S. Tangade, A survey on authentication schemes in VANETs for secured communication, Veh. Commun. 9 (2017) 19–30.
[9] I. Ali, A. Hassan, F. Li, Authentication and privacy schemes for vehicular ad hoc networks (VANETs): a survey, Veh. Commun. 16 (2019) 45–61.
[10] M.N. Mejri, J.B. Othman, M. Hamdi, Survey on VANET security challenges and possible cryptographic solutions, Veh. Commun. 1 (2) (2014) 53–66.
[11] M. Raya, J.P. Hubaux, Securing vehicular ad hoc networks, J. Comput. Secur. 15 (1) (2007) 39–68.
[12] J.M.D. Fuentes, L.G. Manzano, A.I.G. Tablas, J. Blasco, WEVAN-a mechanism for evidence creation and verification in VANETs, J. Syst. Archit. 59 (10) (2013) 985–995.
[13] C. Zhang, R. Lu, X. Lin, P.H. Ho, X. Shen, An efficient identity-based batch verification scheme for vehicular sensor networks, in: IEEE INFOCOM 2008 - the 27th Conference on Computer Communications, 2008, pp. 246–250. Phoenix, AZ.
[14] J.L. Tsai, A new efficient certificateless short signature scheme using bilinear pairings, IEEE Syst. J. 11 (4) (2017) 2395–2402.
[15] What is a Merkle Tree, beginner's guide to this blockchain component, Available: https://blockonomi.com/merkle-tree/. (Accessed 18 May 2019).
[16] Z. Lu, W. Liu, Q. Wang, G. Qu, Z. Liu, A privacy-preserving trust model based on blockchain for VANETs, IEEE Access 6 (2018) 45655–45664.
[17] M. Wazid, A.K. Das, R. Hussain, G. Succi, J. J. P. C. Rodrigues, Authentication in cloud-driven iot-based big data environment: survey and outlook, J. Syst. Archit. 97 (2019) 185–196.
[18] S.S. Al-Riyami, K.G. Paterson, Certificateless public key cryptography, in: Advances in Cryptology-ASIACRYPT 2003, Springer, Berlin, 2003, pp. 452–474. 2612.
[19] A. Shamir, Identity-based cryptosystems and signature schemes, in: Advances in Cryptology Crypto, 1984, p. 4753.
[20] D. He, S. Zeadally, B. Xu, X. Huang, An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks, IEEE Trans. Inf. Forensics Secur. 10 (12) (2015) 2681–2691.
[21] Y. Zhang, R.H. Deng, X. Liu, D. Zheng, Outsourcing service fair payment based on blockchain and its applications in cloud computing, IEEE Trans. Serv. Comput. (2018), doi:10.1109/TSC.2018.2864191.
[22] Y. Zhang, R. Deng, D. Zheng, J. Li, P. Wu, J. Cao, Efficient and robust certificateless signature for data crowdsensing in cloud-assisted industrial iot, IEEE Trans. Ind. Inform. (2019), doi:10.1109/TII.2019.2894108.
[23] X. Huang, Y. Mu, W. Susilo, D.S. Wong, W. Wu, Certificateless signature revisited, in: Information Security and Privacy, Springer, Berlin, 2007, pp. 308–322.
[24] S.K.H. Islam, G.P. Biswas, Provably secure and pairing-free certificateless digital signature scheme using elliptic curve cryptography, Int. J. Comput. Math. 90 (11) (2013) 2244–2258.
[25] D. He, Y. Chen, J. Chen, An efficient certificateless proxy signature scheme without pairing, Math. Comput. Model. 57 (910) (2013) 2510–2518.
[26] N.B. Gayathri, T. Gowri, R.R.V.K. Rao, P.V. Reddy, Efficient and secure pairing-free certificateless directed signature scheme, J. King Saud Univ. - Comput. Inf. Sci. (2018), doi:10.1016/j.jksuci.2018.02.016.
[27] A. Malip, S.L. Ng, Q. Li, A certificateless anonymous authenticated announcement scheme in vehicular ad hoc networks, Secur. Commun. Netw. 7 (3) (2014) 588–601.
[28] A.K. Malhi, S. Batra, An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks, Discrete Math. Theor. Comput. Sci. 17 (1) (2015) 317–338.
[29] S.J. Horng, S.F. Tzeng, P.H. Huang, X. Wang, T. Li, M.K. Khan, An efficient certificateless aggregate signature with conditional privacy-preserving for vehicular sensor networks, Inf. Sci. 317 (2015) 48–66.
[30] J. Li, H. Yuan, Y. Zhang, Cryptanalysis and improvement of certificateless aggregate signature with conditional privacy preserving for vehicular sensor networks, IACR Cryptol. Eprint Arch. 692 (2016).
[31] P. Kumar, S. Kumari, V. Sharma, X. Li, A.K. Sangaiah, S.K.H. Islam, Secure CLS and CL-AS schemes designed for VANETs, J. Supercomput. (2018) 1–23.
[32] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: Advances in Cryptology CRYPTO 2001, Springer, 2001, pp. 213–229. 2139.
[33] D. Boneh, B. Lynn, H. Shacham, Short signatures from the weil pairing, in: Advances in Cryptology ASIACRYPT 2001, Springer, 2001, pp. 514–532. 2248.
[34] M. Barbosa, P. Farshim, Certificateless signcryption, in: Proc. of the 2008 ACM symposium on Information, computer and communications security, 2008, pp. 369–372. Tokyo, Japan.
[35] Z. Zhang, D.S. Wong, J. Xu, D. Feng, Certificateless public-key signature: security model and efficient construction, in: Applied Cryptography and Network Security, Springer, 2006, pp. 293–308. 3989.
[36] Dedicated, Short range communications (DSRC), Available: http://grouper.ieee.org/groups/scc32/dsrc/index.html. (Accessed 20 March 2018).
[37] O. Hyunseo, Y. Chungil, A. Donghyon, C. Hanberg, 5.8 GHz DSRC packet communication system for ITS services, in: Gateway to 21st Century Communications Village. VTC 1999-Fall. IEEE VTS 50th Vehicular Technology Conference (Cat. No. 99CH36324), vol. 4, 1999, pp. 2223–2227. Amsterdam.
[38] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, in: Advances in Cryptology EUROCRYPT 2003, Springer, 2003, pp. 416–432. 2656.
[39] S. Mitsunari, R. Sakai, M. Kasahara, A new traitor tracing, IEICE Trans. Fundam. of Electron. Commun. Comput. Sci. E85-A (2) (2002) 481–484.
[40] H. Du, Q. Wen, Efficient and provably-secure certificateless short signature scheme from bilinear pairings, Comput. Stand. Interfaces 31 (2) (2009) 390–394.
[41] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, 2008, Available: http://www.bitcoin.org/bitcoin.pdf.
[42] A. Dorri, S.S. Kanhere, R. Jurdak, Blockchain in internet of things: challenges and solutions, 2016, Available. arXiv:1608.05187
[43] M. Atzori, Blockchain-based architectures for the internet of things: a survey, SSRN Electron. J. (2017).
[44] Y. Zhang, R.H. Deng, X. Liu, D. Zheng, Blockchain based efficient and robust fair payment for outsourcing services in cloud computing, Inf. Sci. 462 (2018) 262–277.
[45] J. Yu, V. Cheval, M. Ryan, DTKI: a new formalized PKI with no trusted parties, 2014, Available. arXiv:1408.1023
[46] J. Daemen, V. Rijmen, The Design of Rijndael: AESthe Advanced Encryption Standard, Springer, 2002. 235.
[47] MIRACL, cryptographic library: multiprecision integer and rational arithmetic c/c++ library, Available: http://indigo.ie/~mscott/.

Ikram Ali is a Ph.D. candidate in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. He received his master degree in Computer Science (Information Security Management) from Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad Pakistan in 2013. His recent research interests include cryptography and network security.

Gervais Mwitende is now a Ph.D. student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. He received his master degree in Computer Science from National University of Rwanda. His current research interests include cryptography, network security and Internet of Things (IoTs).

Emmanuel Ahene is now a Ph.D. student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. He received his MEng. degree in Computer Science and Technology from University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China in 2016. His recent research interests include cryptography and network security.

Fagen Li is a professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, P.R. China. He received his Ph.D. degree in Cryptography from Xidian University, Xi'an, P.R. China in 2007. From 2008 to 2009, he was a postdoctoral fellow in Future University-Hakodate, Hokkaido, Japan, which is supported by the Japan Society for the Promotion of Science (JSPS). He worked as a research fellow in the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan from 2010 to 2012. His recent research interests include cryptography and network security. He has published more than 100 papers in international journals and conferences. He is a member of the IEEE.