Vehicular Communications 25 (2020) 100255

Contents lists available at ScienceDirect

Vehicular Communications

www.elsevier.com/locate/vehcom

Continuous authentication for VANET

Basker Palaniswamy [a,d], Seyit Camtepe [b,*], Ernest Foo [a], Leonie Simpson [a], Mir Ali Rezazadeh Baee [a], Josef Pieprzyk [b,c]

[a] Information Security Discipline, QUT, Brisbane, Australia
[b] CSIRO Data61, Sydney, Australia
[c] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[d] Information and Communications Technology, University of the Sunshine Coast, Brisbane, Australia

A R T I C L E I N F O

Article history:
Received 20 March 2019
Received in revised form 16 January 2020
Accepted 4 April 2020
Available online 16 April 2020

Keywords:
Secure-vehicular-communication
Security and privacy
VANET authentication
V2I and V2V key exchange
Handoff authentication
Formal verification

A B S T R A C T

Vehicular Ad-hoc Network (VANET) services use a range of information, such as traffic conditions and location information, for safe and convenient driving. Information exchange in VANET happens in Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communication modes. Several V2I and V2V authentication protocol suites are proposed to protect the information from attacks, namely replay, masquerading and man-in-the-middle. This paper identifies critical weaknesses in the protocols proposed for V2I and V2V communication modes and proposes a new protocol suite as a countermeasure. Our protocol suite is composed of driver authentication, V2I and V2V key exchanges, information exchange, offline password change and vehicle complain protocols. Our V2I key exchange protocol facilitates handoff capability to ensure continuous authentication when vehicles move from the coverage of one roadside unit (RSU) to another. The protocol also assures detectability to denial-of-service (DoS). Our V2V key exchange protocol enables vehicles to verify the time-bounded validity of certificates of vehicles and integrity of keys. We use the random oracle model to prove the security of our key exchange protocols and prove various security attributes of the protocols informally. Tamarin tool is used to formally verify the security properties of our driver authentication and key exchange protocols. Performance comparisons show that our driver authentication and key exchange protocols assure lesser computation overhead and more functional attributes than the existing protocols. Simulation performance ensures the fast key dissemination capability of our protocol suite.

1. Introduction

Advancements in vehicular technologies (pedestrian detection system [1], advanced driver assistance system (ADAS) [2] and co-operative adaptive cruise control (CACC) [3])) ensure vehicles to use various information in vehicular ad-hoc network (VANET). The technologies have the potential to assure safer and easier driving than ever before [4]. When vehicles use information from road-side infrastructures (RSUs) and other vehicles, several benefits can be attained by the vehicles. For instance, when vehicles operate in a platoon, improved fuel efficiency can be attained by exploiting information about current driving conditions [5]. Location-based safety information (LBSI) informs drivers about upcoming obstacles or nearest fuel stations [6]. The information is predominantly exchanged to vehicles using vehicle to infrastructure (V2I) and vehicle to vehicle (V2V) communication modes [7].

The information needs protection against adversarial manipulations, such as message alteration and fake message injection, from untrusted sources, e.g. untrusted vehicles and infrastructures [8] [9]. For example, if a malicious vehicle alters the information about a road condition and transmits it to a targeted vehicle, the receiving vehicle may face accidents [10].

The information can be protected by ensuring an authentication protocol suite. Researchers have proposed several authentication protocol suites [11–15] to assure protection for the information. The proposed suites include authentication protocols for driver authentication, and V2I and V2V communication modes. However, the proposed protocols have drawbacks such as lack of resistance to adversarial attacks, excess computation and practicality. The existing V2I key exchange protocols do not offer continuous authentication while disseminating keys to vehicles. That is, the protocols do not support continuous key exchange when vehicles move from

* Corresponding author.
E-mail addresses: basker170889@gmail.com (B. Palaniswamy), Seyit.Camtepe@data61.csiro.au (S. Camtepe), e.foo@qut.edu.au (E. Foo), lr.simpson@qut.edu.au (L. Simpson), mirali.rezazadeh@qut.edu.au (M.A. Rezazadeh Baee), josef.pieprzyk@data61.csiro.au (J. Pieprzyk).

https://doi.org/10.1016/j.vehcom.2020.100255
2214-2096/© 2020 Elsevier Inc. All rights reserved.

the coverage of one roadside infrastructure to another roadside infrastructure (handoff capability [16]). Continuous authentication is needed to disseminate session keys to vehicles quickly. The existing V2V key exchange protocols do not facilitate vehicles to asses the time-bounded validity of certificates at a lower computation overhead, which may result in those vehicles with expired certificates receiving valid session keys. Also, the existing protocols for the driver authentication and V2I and V2V key exchange protocols have not been formally verified. Therefore, our five-folded contributions are as follows:

- We perform a security analysis of Park et al. [17] and Ying et al. [15] protocol suites and identify their weaknesses.
- We present a driver authentication protocol which can resist to known attacks, namely replay, masquerading and man-in-the-middle [18].
- We propose V2I and V2V key exchange protocols. In particular, our V2I key exchange protocol exhibits continuous authentication characteristics. Our V2V key exchange protocol facilitates vehicles to check the time-bounded validity of a certificate of a vehicle and integrity of session keys.
- We perform the security analysis of our key exchange protocols in the random oracle model [19]. We perform formal security verification of the proposed protocol suite using the Tamarin tool.
- We simulate our protocol suite and compare its key dissemination capability with the protocol of Park et al. [17]. We show that our protocol suite also has a fast key dissemination capability.

The outcome of this research benefits vehicle manufacturers and smart city domains. Vehicle manufacturers can follow the design steps in the proposed driver authentication protocol for implementing a driver authentication system. The new protocol suite assures design steps for implementing a secure LBSI service in a smart city. In general, this research helps VANET community on implementing a fast and secure key exchange protocol for V2I and V2V communication modes.

The road map of this paper is as follows. Section 2 overviews the related work. Section 3 presents the preliminaries. In Section 4, we analyse the security of the Park et al. and Ying et al. protocol suites. We describe our protocol suite in Section 5. Section 6 presents formal security proofs in the random oracle model and informal security analysis of our protocol suite. Formal security verification is presented in Section 7. Sections 8 provides a performance comparison of our protocol suite with relevant protocols. Section 9 presents the comparison of the simulation performance of our protocol suite with the protocol suite of Park et al. [17], and Section 10 concludes our work.

2. Related work

We identify two categories in the existing proposals to secure V2I and V2V communication modes in VANET. They are V2I and hybrid schemes that incorporate both V2I and V2V communication modes. In the following subsections, we analyse the existing V2I and hybrid schemes with a specific focus on their security and efficiency.

2.1. V2I schemes

The existing protocols for the V2I communication mode has disadvantages regarding excess computation overhead and impracticality. The V2I protocols are infrastructure dependent, and they are executed among three parties: a vehicle, an RSU and a trusted server/service provider. Some of the protocols are simplified into two-party protocols by merging an RSU with a trusted server/service provider. For example, Huang et al. [12] have proposed an anonymous batch authenticated and key agreement protocol for value-added service, which is referred to as ABAKA. ABAKA removes the bottleneck of verification delay by employing batch verification procedure. As a result, multiple vehicle authentications and vehicle key agreements are possible in the protocol. Further, the protocol preserves the privacy of vehicles. However, ABAKA establishes pairwise keys to secure communication between vehicles and a service provider, and it demands time synchronization between vehicles and the service provider.

Chim et al. [13] have proposed a protocol suite, which is referred to as SPECS. SPECS offers an efficient batch verification mechanism and a handshake protocol to share a key between a vehicle, and an RSU and a trusted authority. The protocol establishes group keys among vehicles. The protocol facilitates vehicles to form a group dynamically. The protocol uses digital signatures and also exhibits a significant computation overhead.

Wang et al. [11] have proposed the 2FLIP protocol. 2FLIP applies two-factor (i.e. biometrics and a hardware token) driver authentication with an option to trace every driver. The protocol has a system key update phase. As the protocol derives the current key from the most recent one, there is a risk that all future keys can be compromised once one of the past keys is compromised. This protocol also utilises digital signature algorithm during the key update phase.

The PACP protocol has been proposed by Huang et al. [20] to ensure the privacy of vehicles during authentication. The protocol has three phases: registration, generation and extraction. The protocol allows vehicles to complain about other misbehaving vehicles to a trusted server. Nevertheless, an adversary can make fake complaints about vehicles to the trusted server as the vehicle complain protocol does not require authentication of vehicles.

Horng et al. [14] have analysed and presented two impersonation attacks on the SPECS protocol. In the first impersonation attack, Horng et al. [14] have shown a malicious vehicle can forge the signature for a message on behalf of a group member. In the second attack, Horng et al. [14] have shown that an attacker can forge the signature for a group message. An improved protocol called b-SPECS+ has been proposed in the work [14]. However, to prevent the replay attack, b-SPECS+ forces RSUs to store the pseudo-identities used by vehicles. It should be noted that RSUs are susceptible to hardware tampering and less trusted than a trusted authority [21].

A light-weight group key protocol has been proposed by Ying et al. [15], which is referred to as LGKP. LGKP proposes anonymous light-weight authentication based on a smart card that creates a dynamic login ID for drivers and also facilitates to trace the drivers individually. LGKP proposes an efficient password change phase that does not require the involvement of the trusted authority. The driver authentication protocol does not preserve the privacy of drivers. The security limitations of the protocol suite are presented in Section 4. In general, all the protocols presented above does not support the handoff capability.

Li et al. [22] have proposed a protocol suite, which is referred to as LIAP. LIAP facilitates the handoff capability with less handoff delay. The purpose of LIAP is to establish a shared session key between a vehicle and an authentication server based on the pre-shared login details. LIAP does not require a separate authenticated key exchange protocol for sharing session keys. However, LIAP employs a conventional public-key cryptosystem for performing encryption and decryption operation, which incurs heavy computation overhead.

Zhou et al. [23] have analysed LIAP. Zhou et al. [23] have found that LIAP lacks privacy-preserving property and susceptible to impersonation attack. In response, Zhou et al. [23] have proposed a privacy-aware fast handover authentication protocol, which is referred to as PAFH. Nevertheless, PAFH does not provide formal verification.

2.2. Hybrid schemes

Very few protocol suites address the authentication of V2V communication mode. Zhang et al. [24] have proposed a scalable and robust authentication protocol (SRAP) to prevent malicious vehicles from masquerading as a legitimate vehicle. During V2I and V2V communication, SRAP uses a computationally expensive operation, namely the signcryption algorithm. SRAP maintains group communication between RSU and vehicles. However, in SRAP, V2V communication cannot be established between two vehicles that belong to two different groups as each RSU individually establishes a key exchange with vehicles.

Azees et al. [6] have proposed the EAAP protocol that permits vehicles to communicate in the absence of RSU coverage. The protocol achieves the infrastructure-independent V2V communication by generating a sufficient number of short-term public keys during registration. However, EAAP communication between vehicles allows an adversary to link the short-term public keys because a static dummy ID for the vehicles is used in every session of the protocol. Also, messages transmitted from one vehicle to another are prone to a replay attack as they do not carry timestamps. SRAP is also susceptible to replay attacks.

Park et al. [17] have proposed a privacy-preserving group key protocol that is referred to as FKDP. FKDP offers a V2V group key dissemination, where the validity of a certificate (i.e. time-bounded validity of a public key issued by a trusted server) of a vehicle can be verified by another vehicle without contacting a server. They achieve a V2V key exchange in the absence of the coverage of RSU by enabling a vehicle with an updated key to update remaining vehicles. But their protocol is also prone to attacks. A detailed security analysis of FKDP is presented in Section 4.

2.3. Motivation

The protocols discussed for the V2I communication mode fall into two categories. They are protocols that provide a light-weight authentication which does not have handoff capability and the protocols that support handoff capability with substantial computation overheads. Therefore, we intend to design a light-weight authentication protocol that supports handoff capability and additional functional features for V2I communication mode. There has not been any protocol proposed for the V2V communication mode to exchange group keys without security issues. So we design an authenticated key exchange protocol suite for V2V communication mode that is free from known attacks. We design a privacy-preserving authentication scheme for driver authentication in vehicles.

3. Preliminaries

This section provides an introduction to pairing along with computational intractability assumptions, network and adversary models and security goals and requirements.

3.1. Bilinear mapping

Pairing is a map e with characteristics of simple computations, non-degeneracy and bilinearity [25]. It establishes a relation between cryptographic groups by combining elements of two groups to yield an element in a third group. In elliptic curve, paring is a recent addition which brings in encrypted multiplication capabilities. For example, with pairing, it is possible to check quadratic constraints such as $p \cdot q = \text{r}$ by only knowing $\text{P} = G^p$, $\text{Q} = G^q$ and $\text{R} = G^r$ (p, q and r are scalar values, and P, Q and R are elliptic curve points on three groups, and G is the generating point). This paper uses pairing in the proposed V2V protocol for efficiently checking such quadratic constraints by using elliptic curves.

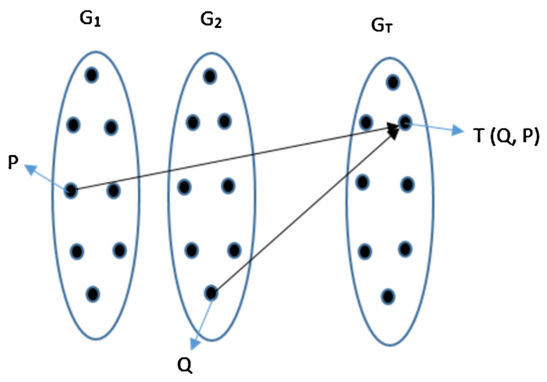

Fig. 1. Ate pairing.

Consider three subgroups $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \cdot)$ defined by three elliptic curves $E(\mathbb{F}(q))$, $E(\mathbb{F}(q^k))$ and $E(\mathbb{F}(q^*))$, respectively. The groups are of prime order q. P and Q are points of the group $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Further, let a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be an admissible bilinear map that satisfies the following properties: bilinearity, non-degenerate and computability [25].

3.2. Ate pairing

The bilinear map can be computed by three types of pairing. They are Weil, Tate and Ate pairings. We consider ate pairing because it is more efficient than other pairings. Let E(F_q) be a collection of points defined by the elliptic curve E(F_q): $y^2 = x^3 + ax + b \mod r$, where $(a, b) \in F_q$ [26]. A pairing-friendly curve is a Barreto-Naehrig (BN) curve, which is represented as E(F_q): $y^2 = x^3 + b \mod r$, where $b \in F_q$. The bilinear map is usually implemented using Weil and Tate pairings. But, for better performance, the map is implemented using Ate pairing [27]. Ate pairing is an optimized version of Tate pairing. Ate pairing is achieved by the Frobenius endomorphism π_q, i.e. π_q: $E \rightarrow E$: $(x, y) \rightarrow (x^q, y^q)$. Now, G_1 and G_2 can be written as $G_1 = E(r) \cap Ker(\pi_q - [1]) = E(F_q)[r]$ and $G_2 = E(r) \cap Ker(\pi_q - [q]) = E(F_q^k)[r]$, where Ker denotes the kernel of π_q and $E(r)$ specifies r-torision which means that entire group of points of order r. The Ate pairing is given by $T(Q, P)^m = f_{r,Q}(P)^{m(q^k-1)/r} = f_{mr,Q}(P)^{(q^k-1)/r}$, where r is not the factor of m, $f_{r,Q} = r(Q) - r(O)$, O is a point at infinity, T is a tate pairing, $m \in Z_p$, $P \in E(r) \cap Ker(\pi_q - [1])$ and $Q \in E(r) \cap Ker(\pi_q - [q])$. We use the same notations as presented in [26]. Fig. 1 presents the Ate pairing. It presents the way a mapping operation is done in a targeted group.

Now, we discus the computational intractability assumptions that are used in our protocols.

Computational intractability 1: Elliptic Curve Discrete logarithmic problem (ECDLP) – Given P and $x \cdot P$ in $E(\mathbb{F}(q))$, where $x \in \mathbb{Z}_p$, it is computationally infeasible to find x in polynomial time.

Computational intractability 2: Elliptic Curve Diffie-Hellman problem (ECDHP) – Given P, $x \cdot P$ and $y \cdot P$ in $E(\mathbb{F}(q))$, where $x, y \in \mathbb{Z}_p$, it is computationally infeasible to compute the product $x \cdot y \cdot P$ in polynomial time.

Computational intractability 3: Elliptic Curve q-Strong Diffie-Hellman problem (ECq-SDHP) – Based on the q-SDH problem [28], we define $ECq - SDHP$ as follows: Given a $(q+3)$-tuple $(t_d, P, Q, x \cdot Q, x^2 \cdot Q, \ldots, x^q \cdot Q) \in \mathbb{Z}_p \times E(\mathbb{F}(q)) \times E(\mathbb{F}(q^k)^{q+1})$, it is computationally intractable to come with a pair $(\delta, \frac{1}{x+t_d \cdot \delta} \cdot P) \in \mathbb{Z}_p \times E(\mathbb{F}(q))$ in polynomial time.

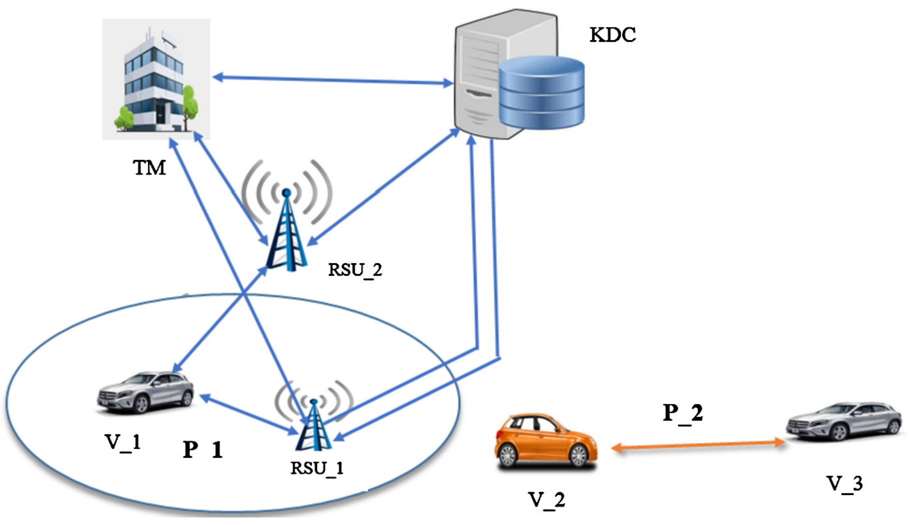

Fig. 2. Network architecture.

3.3. Network architecture

Fig. 2 presents the network architecture of a VANET. It includes a key distribution centre (KDC), trace managers (TMs), vehicles (V_1 to V_3) and RSUs (RSU_1 and RSU_2). The KDC is a trusted authority in the network. It is responsible for the registration of vehicle owner/driver, vehicles and RSUs, generation and distribution of session keys to vehicles and RSUs (P_1 in Fig. 2) and revocation of vehicles. KDC generates a new session key on the following events: a vehicle is newly registered, on the expiry of a public key, a vehicle is revoked, or the current session key is expired. TMs are distributed throughout VANET [29]. In practice, TM is the law authority [30]. The role of TMs is to trace malicious activities and update vehicles with the revocation list. The malicious activities during P_1 and P_2 are reported to a TM by vehicles. Upon receiving complaints from vehicles, TM communicates with KDC to report the malicious vehicle. Depending on the decision, KDC updates TMs with the revocation list such that vehicles can be notified to update the revocation list. Vehicles are equipped with an on-board unit (OBU) and a smart card reader. OBUs, RSUs and TMs contain a tamper-resistant hardware security module [31], where long-term secrets of the vehicle are stored. Every vehicle has a unique identity, such as a chassis number. Vehicles can be operated manually or drive autonomously. Every RSU possesses a unique serial number. RSUs assist vehicles by providing LBSIs.

Vehicle driver/owner knows the password of the vehicle and holds a smart card, which stores the registration credentials. For simplicity, we consider a driver and an owner are the same. V2I and V2V interactions can be carried out using dedicated short-range communication (DSRC) [32]. RSUs and KDC can communicate using a high bandwidth wired channel (e.g. optic link). Similarly, KDC and TMs can communicate using an optic link.

We use the following assumptions about entities, communication channel and mobility of vehicles.

- *B1. Entity assumptions:* Being a trusted authority, KDC cannot be compromised. But vehicles and RSUs are susceptible to compromise. Nevertheless, long-term secrets are stored in the vehicles and RSUs cannot be disclosed.
- *B2. Channel assumptions:* The wireless channel between vehicles and RSUs and the wired channel between RSUs and KDC in V2I communication are susceptible to compromise. The wireless channel between two vehicles in V2V communication is also susceptible to compromise.
- *B3. Mobility assumptions:* There exist some vehicles in the region that is not covered by RSUs. For instance, V_2 and V_3 in Fig. 2 present the location of vehicles in the region. Some vehicles may cross the boundary between two RSUs. For example, V_1 crosses the boundary (ellipsed portion) of RSU_1 and enters into the boundary of RSU_2 in Fig. 2.

3.4. Adversarial capabilities

An adversary is a probabilistic polynomial-time algorithm ($\mathcal{A}$) with complete control over messages. Precisely, it can eavesdrop, inject, delete, spoof and replay messages. The adversary can tamper with hardware units of either OBUs of vehicles or RSUs to acquire stored secrets [21]. This means that the adversary can acquire secret random numbers and session keys that are used by vehicles and RSUs during previous key exchanges. However, the adversary cannot acquire the long-term secrets of KDC. The adversary can tamper with the smart-card and recover all the stored credentials, and it can accurately guess the password in polynomial time.

3.5. Security goals and requirements

An authenticated key exchange protocol (AKEP) should satisfy the following security goals:

- *D1. Mutual entity authentication* allows two communicating entities (parties) to verify their identities (through acquisition or collaborative evidence).
- *D2. Explicit key authentication* convinces parties that they are the only ones that hold a correct session key.
- *D3. Key confirmation (KC)* is guaranteed if a party proves the possession of a session key.
- *D4. Key freshness* guarantees that the current session key is chosen at random and independently [33,34].
- *D5. Key integrity* holds if any unauthorised alteration of keys is detectable [33].
- *D6. Forward secrecy (FS)* holds if the disclosure of long-term secrets of a party does not reveal any session keys to $\mathcal{A}$ [34].
- *D7. Known-key secrecy (KKS)* holds if the exposure of any session key does not help $\mathcal{A}$ in attaining the present session key [35,34].

AKEP is intended to satisfy conditional anonymity and unlinkability to ensure the privacy of vehicles and vehicle drivers [11].

- *D8. Conditional anonymity* of a vehicle is achieved if KDC alone has the resolution to the pseudonyms used by vehicles [36].
- *D9. Unlinkability* of a vehicle is achieved if an adversary cannot link multiple pseudonyms used by a vehicle when observing its communication [36].

AKEP should be resilient against the following attacks: password guessing, lost smart card, replay, masquerading and MITM, and it should detect denial of service (DoS) [37].

4. Analysis of Park et al. and Ying et al. protocol suites

We present our analysis of the protocol suite of Park et al. [17] and Ying et al. [15]. We use the same notations as used in the works [17,15].

4.1. Park et al. protocol suite

V2I group key transmission protocol is susceptible to MITM attack, and the protocol does not preserve the privacy of vehicles. V2V group key transmission protocol lacks KKS.

4.1.1. V2I

KDC updates keys via RSUs. The objective of this protocol is to exchange keys between vehicles and a trusted server. A vehicle should possess the most recent key to execute the protocol. The protocol is presented below.

Vehicle 1) *Send the identity (ID) of the vehicle as plain text*;
RSU/KDC 2) *Verify the expiry time of the public key of the vehicle*; 3) *Check the version of the group key possessed by the vehicle*; 4) *Encrypt new group key using the public key of the vehicle*; 5) *Send the encrypted group key*;
Vehicle 6) *Decrypt the cipher text using the private key of the vehicle to obtain the group key*;

<u>Security Analysis</u>: Let the version of the group key possessed by the vehicle be GK_2. Adversary $\mathcal{A}$ starts an attack by sending the ID of the target vehicle (Step 1) and continues with the following steps:

$\mathcal{A}$ 1) *Replay* ID;
RSU/KDC 2) *Verify the expiry time of the public key of the vehicle*; 3) *Check the version of the group key possessed by the vehicle*; 4) *Encrypt* GK_3 *using the public key of the vehicle* (k); 5) *Send* $\{GK_3\}k$;
$\mathcal{A}$ 6) *Store* $\{GK_3\}k$;
Vehicle 7) *Send* ID;
RSU/KDC 8) *Verify the expiry time of the public key of the vehicle*; 9) *Check the version of the group key possessed by the vehicle*; 10) *Encrypt* GK_4 *with* k; 11) *Send* $\{GK_4\}k$;
$\mathcal{A}$ 11) *Block* $\{GK_4\}k$; 12) *Send* $\{GK_3\}k$;
Vehicle 13) *Decrypt* $\{GK_3\}k$; 14) *Accept* GK_3;

Consequently, the vehicle holds group key GK_3 while KDC's version of the group key is GK_4. In other words, the vehicle cannot communicate with the group as it uses an old key. The attack is possible if the vehicle leaves the network at the end of a day and joins the network in the following morning. The duration is sufficient for the key update to happen twice. It should be noted that the privacy of the vehicle is also violated since the identity of the vehicle is explicitly used to update keys. Hence, the protocol does not preserve the conditional anonymity and unlinkability of vehicles.

4.1.2. V2V

The objective of the protocol is to perform the key update between vehicles in the absence of RSU coverage. The keys are exchanged over a confidential channel which is established using the existing common group key. The party requesting the key update proves their identity to the party supplying the new key, using an asymmetric key-based challenge-response mechanism. The steps below present the protocol.

Alice 1) *Request using group name*; 2) *Open secure channel*;
Alice 3) *Send encrypted subscription information*;
Bob 4) *Send challenge*;
Alice 5) *Compute response for the challenge*; 6) *Send the response*;
Bob 7) *Verify the response* 8) *Send encrypted GK*;
Alice 9) *Verify GK*;

<u>Security Analysis</u>: This protocol lacks KKS. As the secure channel is opened based on the current group key, knowledge of a previous session key can help $\mathcal{A}$ to acquire the future session key just by eavesdropping the communications.

4.2. Ying et al. protocol suite

A trusted server updates keys of vehicles after verifying the authenticity of vehicles. For the nomenclature, the reader is referred to the work of Ying et al. [15]. LGKP is described below.

Step 1: The dynamic login identity $DIDV_{i,j}$ is created as follows:

- Compute $K = N_i \oplus H_0(PW_i \parallel n_i)$.
- Compute $C_1 = y^{H_0(ID_{V_i})} \bmod p$.
- Compute $DIDV_{i,j} = H_0(C_1 \parallel H_0(ID_{V_i} \parallel n_j))$, where n_j is a random number.
- Compute $CV_i = H_0(DIDV_{i,j} \parallel k)$
- Send $< DIDV_{i,j}, CV_i, n_j, T_{V_i} >$ to R_i, where T_{V_i} is the time stamp.

Step 2: When ith RSU receives the login message at time T, it will act as follows:

- Check $(T - T_{V_i}) \le \delta T$.
- Compute $DIDR_i = DIDV_{i,j} \oplus H_0(ID_{R_i})$.
- Send $< DIDR_i, CV_i, n_j, T_{R_i} >$ to trusted authority.

Step 3:Trusted authority carries out the following steps:

- Check $(T_1 - T_{R_i}) \le \Delta T$.
- Check $C_1 = y^{PVID_{V_i}} \bmod p$.

Table 1
Nomenclature.

Notation	Meaning
t_d t_c	Vehicle's departure time Current time
δ, ε	KDC's secrets
Y & D	Helper values, where $Y = \epsilon \cdot Q$ and $D = \delta \cdot Q$
GK_{part1},GK_{part2}	Group key part 1 and part 2
λ	Seed value for generating the group key
ver_{gk}	Version of group key
$f()$	Identity function
$\{\}$	Encryption function
$GK = f(GK_{part_1}, GK_{part_2}, ver_{gk})$	Group key
$ID_{Vehicle_i}$	Identity of the ith vehicle
SID	Anonymous ID of a vehicle
L_{RSU_i}	Long-term secret between ith RSU and KDC
L_{TM_i}	Long-term secret between ith TM and KDC
L_{RTM_i}	Long-term secret between ith RSU and TM
RL_i	ith revocation list generated by KDC
L_{VK}	Long-term secret between ith vehicle and KDC
K_c	Long-term group secret among vehicles
r, s, d	Random integers
$\oplus$	Bitwise XOR
$H(.): \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^k$	Hash function (k is a constant)
$IBF: Z_p \rightarrow \{0,1\}^c$	Integer to binary conversion function
$IBF^{-1}: \{0,1\}^c \rightarrow Z_p$	Inverse IBF function
$ID_{Vehicle_i}$ & ID_{RSU_i}	Identity of the ith vehicle and RSU, respectively
PWD_i	Password chosen by the ith vehicle
$Cert^*_{i,Alice(i),Bob(i)}$	ith certificate of Alice and Bob
$PKV_{Alice(i),Bob(i)}$	ith public key of Alice and Bob
ΔT, T_c & T_r	Accepted time delay, current and received time

- Compute $DIDV_{i,j} = H_0(C_1 \parallel PVID_{V_i} \parallel n_j)$.
- Compute $H_0(ID^*_{R_i}) = DIDV^*_{i,j} \oplus DIDR_i$ and check $H_0(ID^*_{R_i}) = H_0(ID_{R_i})$.
- $k = H_0(x \parallel PVID_{V_i} \parallel T_{reg})$
- Check $CV_i = H_0(DIDV_{i,j} \parallel k)$, $K_s = H_1(C_3)$, $M_i = E_{K_s}(K_c \oplus k)$, where K_c is a common random number.
- Send $< C_3, M_i, T'_{V_i} >$

Step 4: RSU does the following:

- $(T_2 - T'_{R_i}) \leq \Delta T$
- Send the message $< C_3, M_i, T'_{V_i} >$

Step 5: Vehicle performs the following steps:

- $(T_3 - T'_{V_i}) \leq \Delta T$
- Compute $C_3 = H_1(k \parallel H_0(ID_{V_i}))$ and check $C^*_3 = C_3$
- $K^*_s = H_1(C^*_3)$
- Decrypt M_i using K^*_s and store K_c

Security Analysis: The protocol is vulnerable to a masquerading attack. The following attack explains the way any vehicle gets the secret key k of another registered vehicle. The protocol is a group key protocol, and the session key K_c is known to all group members. Knowing K_c, a malicious group member can compute $K_s = H_1(C_3)$ (refer sixth bulletin in Step 3) as C_3 is a public value. It can decrypt $M_i = E_{K_s}(K_c \oplus k)$. From $K_c \oplus k$, the session key k of any other group member can be recovered as $K_c \oplus k \oplus K_c$. It is noted that the arbitrary vehicle can complete the whole protocol until Step 5. Hence, the protocol lacks the known key secrecy. If anyone of the past session keys is leaked from the recorded message, the current session key k can be acquired as mentioned above.

5. Our protocol suite

Our protocol suite consists of ten protocols: vehicle registration protocol (VRP), RSU registration protocol (RSURP), TM registration protocol (TMRP), vehicle user authentication protocol (VUAP), tri-party key dissemination protocol for V2I communication (TV2I), key dissemination protocol for V2V communication (KV2VP), a protocol for exchanging messages (PEM), offline password change protocol (OPCP) and vehicle complain protocol (VCP). In particular, VUAP, TV2IP, PEM and VCP are symmetric-key protocols and KV2VP is an asymmetric key protocol.

The protocol starts with TMs registration (TMRP), vehicle owners registration (VRP) and RSUs registration (RSURP). Fig. 3 presents the steps involved in VRP and VUAP. O_1 in Fig. 3 shows the registration of an owner with KDC. Vehicle and its owner/driver carries out VUAP to mutually authenticate each other before connecting to VANET. O_2 in Fig. 3 shows the steps involved in VUAP. Upon successful execution of VUAP, vehicles follow TV2I to update keys from KDC. Update of keys between vehicles can be done by following KV2VP. PEM is performed by RSUs and vehicles to exchange information, such as LBSIs, information about current traffic status, etc. Misbehaving vehicles or RSUs during the protocol actions can be intimated to KDC by following VCP.

Table 2 presents goals of the protocols, parties and specifies the type of communication channel used. We follow the notations from Table 1. Before presenting the protocols, we describe the procedure for generating keys by KDC.

5.1. Group key generation procedure and setup (GGPS)

The objective of the protocol is to generate group keys. GGPS assures the key integrity, i.e. vehicles can check the integrity of keys and ensure that the keys are indeed generated by the KDC. KDC generates a group key as follows:

$$ver_{gk} = ver_{gk} + 1$$

$$K = H(\lambda \parallel ver_{gk} \parallel K_c) \text{ where } \lambda \in_R \mathbb{Z}_p$$

$$GK_{part2} = K, \; GK_{part1} = \frac{1}{K \cdot \epsilon} \cdot P \text{ for } \frac{1}{K \cdot \epsilon} \neq (0,1)$$

$$GK = f(GK_{part1}, \; GK_{part2}, \; ver_{gk})$$

Consuming GK_{part1}, GK_{part2} and ver_{gk} as separate values, vehicles can check the integrity of the key as follows:

$$e(GK_{part1}, GK_{part2} \cdot Y) \iff e(P, Q)$$

KDC publishes the description of the groups G_1, G_2 and G_T. Then, it announces two points P and Q that generate G_1 and G_2, respectively. It makes the following public: e, $f()$ and ψ, where e is a bilinear mapping, $f()$ is an identity function and ψ is the isomorphism between G_1 and G_2. Finally, it publishes the helper values (ref. Table 1).

A group key is generated if any of the following events occurs in VANET: (1) a new vehicle or an RSU is registered, (2) certificate validity of a registered vehicle has expired, (3) a vehicle is revoked and (4) any of the registered RSUs is replaced due to repair.

Table 2
Protocols, objectives and involved parties.

Phase	Protocol	Channel	Parties	Primary security goal
Registration	VRP	Secure*	Vehicles and KDC	Share secret elements
Registration	RSURP	Secure*	RSUs and KDC	Share secret elements
Registration	TMRP	Secure*	TMs and KDC	Share secret elements
User authentication	VUAP	Insecure smart-card	Driver/owner, vehicle and	Authenticate owner/driver
Key dissemination	TV2I	Insecure	Vehicle, RSUs and KDC	Exchange session key
Key dissemination	KV2VP	Insecure	Vehicles	Exchange session key
Message exchange	PEM	Insecure	Vehicles and RSUs	Exchange LBSIs
Password change	OPCP	Secure*	Driver/owner and KDC	Exchange PWD and smart-card credentials
Vehicle complain	VCP	Insecure	Vehicles, RSUs, TM and KDC	Making complaints of misbehaving vehicles

* Initial registration is achieved at a physical proximity.

5.2. Vehicle registration protocol (VRP)

This protocol aims to register vehicles with a trusted authority, KDC. The registration procedure between the ith vehicle and KDC is presented below. A vehicle owner chooses a password and submits it to KDC. The owner gives an identity of the vehicle, which is the chassis number (Step 1) and provides their contact number, date of birth (DoB) and primary and secondary email IDs. KDC generates credentials for the vehicle (Step 3). For the fixed departure time (t_d) of the vehicle, KDC generates certificates ($Cert_i^*$) and corresponding public keys (PKV_i^*) sufficient for the vehicle by varying x (s.t. x^*). KDC sends the credentials to the vehicle (Step 5). The vehicle stores the credentials (Steps 6 and 7).

```
**Vehicle_i**
1) Send $ID_{Vehicle_i},\ PWD_i$;

**KDC**
2) Choose $x^*, t_d, d, L_{VK} \in_R Z_p$;  Ensure $x^* + \delta t_d \neq (0,1)$;
3) Compute $PKV_i^* = H(x^* \parallel \delta \parallel ID_{Vehicle_i}) \cdot Q$,
   $Cert_i^* = \frac{1}{H(x^* \parallel \delta \parallel ID) + \delta t_d} \cdot P,\ m = H(PWD_i)$,
   $n = H(m \parallel \delta \parallel ID_{Vehicle_i}),\ SID = H(\delta \parallel d \parallel ID_{Vehicle_i})$,
   $y = m \oplus n,\ C = H(y)$;
4) Maintain a tuple
   $< SID, H(x^* \parallel \delta \parallel ID)^*, m, n, C, PKV_i^*, Cert_i^*, ver_{gk} >$;
5) Send $H(x^* \parallel \delta \parallel ID)^*, SID, C, n, m, PKV_i^*, Cert_i^*, t_d, GK$;

**Vehicle_i**
6) Store $H(x^* \parallel \delta \parallel ID)^*, PKV_i^*, Cert_i^*, m, C, ver_{gk}$
   in the tamper − resistant module;
7) Store $n, SID$ in the smart card;
```

5.3. RSU registration protocol (RSURP)

The following protocol registers RSUs with KDC. KDC delivers L_{RSU_i} and GK to RSU_i. KDC maintains a list of registered RSUs based on its geographical location.

```
**RSU_i**
1) Send $ID_{RSU_i}$;

**KDC**
2) Choose $L_{RSU_i} \in_R \{0,1\}^*$;
3) Send $L_{RSU_i}, GK, L_{RTM_i}$;

**RSU_i**
4) Store $L_{RSU_i}, L_{RTM_i}$ in tamper − resistant module;

**KDC**
5) Maintain a tuple $< Id_{RSU_i}, L_{RSU_i}, ver_{gk} >$;
```

5.4. TM registration protocol (TMRP)

The goal of this protocol is to register TMs with the KDC. The below protocol explains the steps involved during the registration of ith TM with KDC. KDC delivers L_{TM_i} and RL_i to TM_i. KDC maintains a list of registered TMs and its long-term secret. The execution of this protocol is less frequent as TMs are fixed compared to vehicles.

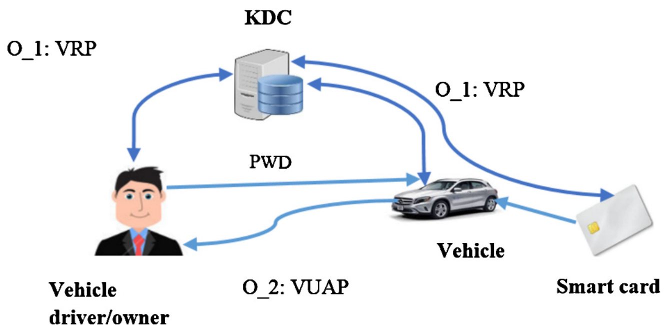

Fig. 3. VRP & VUAP.

```
**TM_i**
1) Send $ID_{TM_i}$;

**KDC**
2) Choose $L_{TM_i} \in_R \{0,1\}^*$;
3) Send $L_{TM_i}, L_{RTM_i}, RL_i$;

**TM_i**
4) Store $L_{TM_i}, L_{RTM_i}$ in tamper − resistant module;

**KDC**
5) Maintain a tuple $< Id_{TM_i}, L_{TM_i}, RL_i >$;
```

5.5. Vehicle user authentication protocol (VUAP)

The objective of this protocol is to authenticate vehicle user/owner. The protocol presented below preserves the conditional anonymity and unlinkability of vehicles. A vehicle user/owner enters a password after inserting a smart card (Step 1). The smart card performs the computations (Step 2), where T_c is the current time, and sends the message (Step 3). The vehicle performs two checks (Step 4). The first check is for the time delay, where T_r is the time the message is received to prevent the replay attack. In the second check, the vehicle checks validity of $H(H(y) \parallel T_c)$ by comparing it with $H(C \parallel T_c)$, where C is fetched from the tamper-resistant module. Unless the two checks are valid, the vehicle aborts the authentication and blocks the user from accessing the vehicle (Step 5).

```
**Vehicle User**
1) Enter $PWD_i$;
2) Compute $m = H(PWD_i), y = m \oplus n, H(H(y) \parallel T_c)$;
3) Send $T_c \parallel H(H(y) \parallel T_c)$;

**Vehicle_i**
4) Check $|T_r - T_c| \le \Delta T,\ H(H(y) \parallel T_c) == H(C \parallel T_c)$;
5) Authenticate the vehicle user;
```

5.6. Tri-party key dissemination protocol for vehicle to infrastructure communication (TV2I)

The objective of this protocol is to exchange keys to vehicles from a trusted authority via RSUs. The parties involved in this protocol are vehicles, RSUs and KDC. The TV2I protocol below depicts steps involved between the ith vehicle and KDC when the vehicle crosses the boundary of ith RSU to either $(i-1)$th or $(i+1)$th RSU. The vehicle forwards M and σ_1 to RSU_i for proving its identity to KDC (Step 3). The RSU constructs U and (σ_2) (Step 5) to prove its authenticity, and it forwards the message to KDC (Step 6). KDC verifies the tag of the RSU, matches SID with the stored SID, checks the validity (t_d) of the vehicle and the version of the key (ver_{gk}) held by the vehicle and verifies the tag of the vehicle (Step 7). KDC aborts authentication if either one of the cases occurs: the tags are invalid, the validity of the vehicle is expired and version of the key matches with the current version. KDC recovers random numbers that are sent by the vehicle and the RSU (Step 9), and it performs Step 10. The tags in Step 10 are computed based on the request received from the RSU. That is, if the ith RSU delivers the message in Step 6, then the tags for the ciphertexts will be additionally computed for $(i-1)$th and $(i+1)$th RSU. The two additional tags are generated to uninterruptedly deliver the message to the vehicle regardless of its direction of mobility. The vehicle can acquire the key and construct the next session identifier (SID^*) by following Step 14. As the key confirmation step, the vehicle computes a tag (Step 15), and it forwards to the RSU. Now, the vehicle is assumed to be on the boundary of new RSU (i.e. either $(i-1)$th or $(i+1)$th. The RSU computes a tag, and it forwards the message to KDC (Step 17 and 18, respectively). KDC verifies the tags and updates the tuple (Step 19 and 20). It should be noted that if any of the verifications or decryptions fail, then the concerned party aborts the protocol.

```
Vehicle_i
1) $S \in_R \{0,1\}^*,\ T = H(L_{VK} \parallel GK),\ B = S \oplus T \oplus L_{VK} \oplus GK;$
2) $\sigma_1 = H(SID \parallel B \parallel L_{VK}),\ M = SID, B;$
3) $Send\ M, \sigma_1;$

RSU_i
4) $R \in \{0,1\}^*,\ U = H(L_{RSU_i} \parallel GK), Q = R \oplus L_{RSU_i} \oplus U \oplus GK;$
5) $\sigma_2 = H(Q \parallel M \parallel \sigma_1 \parallel RSU_i \parallel L_{RSU_i});$
6) $Send\ M, RSU_i, Q, \sigma_1, \sigma_2;$

KDC
7) $Verify\ \sigma_2,\ Match\ < SID, ver_{gk}, t_d >,\ Verify\ \sigma_1;$
8) $Check\ the\ tuple\ < ID_{RSU}, \ldots, ver_{gk} >;$
9) $S = B \oplus T \oplus L_{VK} \oplus GK,\ R = Q \oplus U \oplus L_{RSU_i} \oplus GK;$
10) $Choose\ r \in_R \{0,1\}^*,\ \zeta_S = \{GK^* \parallel r \parallel SID\}S,$
    $\zeta_R = \{GK^*\}R,\ \sigma_3 = H(\zeta_S \parallel \zeta_R \parallel L_{RSU_i}),$
    $\sigma_3^* = H(\zeta_S \parallel \zeta_R \parallel L_{RSU_{i-1}}),\ \sigma_3^{**} = H(\zeta_S \parallel \zeta_R \parallel L_{RSU_{i+1}});$
11) $Send\ \zeta_S,\ \zeta_R,\ \sigma_3,\ \sigma_3^*,\ \sigma_3^{**};$

RSU_(i-1),i,(i+1)
12) $Verify\ \sigma_3^*, \sigma_3, \sigma_3^{**},\ Decrypt\ \zeta_R;$
13) $Send\ \zeta;$

Vehicle_i
14) $Decrypt\ \zeta, SID^* = H(r \parallel SID \parallel GK);$
15) $\sigma_4 = H(\zeta \parallel SID^* \parallel L_{VK});$
16) $Send\ \sigma_4;$

RSU_(i-1)/i/(i+1)
17) $Compute\ \sigma_5 = H(\sigma_4 \parallel \sigma_3^{/*/**} \parallel L_{RSU_{(i-1),i,(i+1)}});$
18) $Send\ \sigma_4,\ RSU_{i-1/i/i+1},\ \sigma_5;$

KDC
19) $Verify\ \sigma_5,\ Compute\ SID^* = H(r \parallel SID \parallel GK);$
20) $Verify\ \sigma_4,\ Update\ < SID^*, \ldots, ver_{gk}^* >\ in\ the\ tuple;$
```

5.7. Key dissemination protocol for V2V communication protocol (KV2VP)

The purpose of this protocol is to exchange keys from Bob to Alice. To transfer an updated key, Alice needs to prove the validity of the certificate (t_d<t_c) to Bob. Also, Bob needs to convince Alice that his departure time is not expired. Once the validity of Bob and Alice is mutually verified, Alice verifies the integrity of the key to avoid a key injection from Bob. As outlined below, Alice picks up one of her public keys and the corresponding certificate (Step 1). She performs the steps until 4 to prove her validity and authenticity. Bob performs the verification (Step 5), and he checks the departure time of Alice (Step 6). If anyone of the verifications fails, Bob aborts the authentication. Bob takes one of his public keys, and he constructs a session key (Step 7). Then, Bob performs Step 8 and Step 9. Alice performs checks similar to Bob (Step 10 and 11). She constructs a session key to decrypt the key (Step 12) and verifies the integrity of the session key (Step 13). She further constructs a tag to finish the key confirmation step (Step 13), and she forwards the tag to Bob (Step 14). Bob checks the tag (Step 15). If any of the verification fails, then the protocol is aborted.

```
Vehicle_Alice
1) $Take\ Cert^*_{Alice(i)}\ \&\ PKV^*_{Alice(i)};$
2) $Choose\ a \in_R \mathbb{Z}_p,\ a \cdot PKV_{Alice(i)};$
3) $\sigma_1 = H(a \cdot PKV_{Alice(i)} \parallel Cert^*_{Alice(i)}, \parallel PKV^*_{Alice(i)} \parallel t_d \parallel GK);$
4) $Send\ ver_{gk},\ a \cdot PKV_{Alice(i)}, t_d,\ Cert^*_{Alice(i)},\ PKV_{Alice(i)},\ \sigma_1;$

Vehicle_Bob
5) $Verify\ \sigma_1;$
6) $Check\ e(Cert^*_{Alice(i)}, PKV^*_{Alice(i)} + t_d \cdot Y) \iff e(P, Q);$
7) $SK = a \cdot PKV_{Alice(i)} \cdot PKV_{Bob(i)}$
8) $\{GK^*, \lambda, ver_{gk}\}SK,$
   $\sigma_2 = H(\{GK^*, \lambda, ver_{gk}\}SK \parallel Cert^*_{Bob(i)} \parallel PKV_{Bob(i)} \parallel \sigma_1 \parallel GK);$
9) $Send\ \{GK^*, \lambda, ver_{gk}\}SK,\ t_d,\ Cert^*_{Bob(i)},\ PKV_{Bob(i)},\ \sigma_2;$

Vehicle_Alice
10) $Verify\ \sigma_2;$
11) $Check\ e(Cert^*_{Bob(i)}, PKV^*_{Bob(i)} + t_d \cdot Y) \iff e(P, Q);$
12) $SK = a \cdot PKV_{Alice(i)} \cdot PKV_{Bob(i)}$
13) $Verify\ GK^*,\ \sigma_3 = H(GK^* \parallel \sigma_2 \parallel SK);$
14) $Send\ \sigma_3;$

Vehicle_Bob
15) $Verify\ \sigma_3;$
```

5.8. Protocol for exchanging message (PEM)

The prime objective is to exchange LBSI. Every LBSI message (LBSIMSG) is exchanged with vehicles by ith RSU as $ID \parallel LBSIMSG \parallel T_c \parallel H(ID \parallel LBSIMSG \parallel GK \parallel T_c)$, where ID for each message is generated as $r \in_R \mathbb{Z}_p$, $H(r \parallel L_{RSU_i})$ and $ID = r \parallel H(r \parallel L_{RSU_i})$ and T_c is the current time. The parties involved in this protocol are vehicles and RSUs. The message is broadcasted with the time stamp to prevent the replay attack. It should be noted that when messages are transmitted from one vehicle to another vehicle, the same format can be utilised by computing the $ID = H(m \parallel r)$.

5.9. Offline password change protocol (OPC)

The purpose of this protocol is to change the password in offline mode. The parties involved in the protocol are vehicle owner and KDC. The owner contacts KDC through mobile or e-mail to change the password. KDC ensures the legitimacy of the owner sufficiently, e.g. by asking DoB and the ID number. Then, the owner gives a new password (PWD_i^*). KDC computes m^*,n^*, y^* and C^* as follows: $m^* = H(PWD_i^*)$, $n^* = H(m^* \parallel \delta \parallel ID_{Vehicle_i})$, $y = m^* \oplus n^*$ and $C^* = H(y^*)$ and delivers the generated credentials to the owner. The owner stores n^*, and y^* and C^* in the smart card and the tamper-resistant module, respectively. The owner confirms KDC about the storage.

5.10. Vehicle complain protocol (VCP)

The objective of this protocol is to complain about a misbehaving vehicle in KV2VP and PEM. The parties involved in the protocol are vehicle, RSU and TM. Vehicles make complaints on two instances. When a benign vehicle tries to get a key after the expiry of its departure time. When a benign vehicle tries to inject an invalid or old key. The misbehaviour in PEM could be injections or replays of messages. Message in Step 1 can contain either the step 4 and 9 of KV2VP or the message in PEM depending on the misbehaviour. The vehicle constructs a tag, and it forwards the message, tag and SID, to RSU_i (Step 3). RSU_i constructs a tag for the received message, and it forwards the received message, the constructed tag and its SID, to TM (Steps 3 and 4, respectively). TM forwards the message to KDC after constructing σ_3. KDC verifies all the tags (Step 7). KDC executes the steps 8 and 9. If the misbehaviour is identified in KV2VP, KDC can identify the misbehaving vehicle by identifying the certificate. If the misbehaviour is in PEM, KDC can compute $H(r \parallel L_{RSU_i})$ and $H(m \parallel r)$ to trace RSUs and vehicles, respectively. KDC updates the revocation list (Step 10), constructs a tag (Step 11) and forwards to TMs (Step 12). TMs execute Step 13, and it forward the revocation list to RSUs (Step 15) after performing Step 14. RSUs perform the steps from 16 to 18. Vehicles accept the revocation list if the verification is successful for σ_6 (Step 19 and 20).

```
Vehicle_i
1) $Message$;
2) $\sigma_1 = H(Message \parallel SID \parallel L_{VK})$;
3) $Send\ SID,\ Message,\ \sigma_1$;
RSU_i
3) $\sigma_2 = H(Message \parallel SID \parallel \sigma_1 \parallel L_{RSU_i})$;
4) $Send\ SID,\ Message,\ RSU_i,\ \sigma_1,\ \sigma_2$;
TM
5) $Compute\ \sigma_3 = H(SID \parallel Message \parallel \sigma_1 \parallel \sigma_2 \parallel L_{TM_i})$;
6) $Send\ SID,\ Message,\ \sigma_1,\ \sigma_2,\ \sigma_3$;
KDC
7) $Verify\ \sigma_1,\ \sigma_2,\ \sigma_3$;
8) $Extract\ ID\ of\ the\ vehicle\ from\ the\ message$;
9) $Check\ the\ threshold$;
10) $Update\ RL_i$;
11) $Compute\ \sigma_4 = (RL_i \parallel L_{TM_i})$;
12) $Send\ RL_i,\ \sigma_4$;
TM
13) $Verify\ \sigma_4$;
14) $Compute\ \sigma_5 = H(RL_i \parallel L_{RTM_i})$;
15) $Send\ ID_{TM_i},\ RL_i\ \sigma_5$;
RSU_i
16) $Verify\ \sigma_5$;
17) $Compute\ \sigma_6 = H(RL_i \parallel GK)$;
18) $Send\ RL_i\ \sigma_6$;
Vehicle_i
19) $Verify\ \sigma_6$;
20) $Accept\ RL_i$;
```

6. Security analysis

In this section, we present formal security analysis of TV2I and KV2VP protocols. Informal analysis of the security protocols is also presented.

6.1. Formal security analysis of TV2I and KV2VP

The TV2I and KV2VP protocols use long-term secrets (L_{VK}, L_{RSU_i}, $L_{RSU_{(i-1)}}$, K_c, δ and $L_{RSU_{(i+1)}}$) and (x_A, x_B, ϵK_c and δ), respectively, to establish a session key GK^* in a given session. The formal security analysis applies the random oracle model [19] with the adversarial capabilities mentioned in the work [35]. We use the following assumptions. A generic party $\mathcal{G}$ is introduced on behalf of KDC, the RSU_i and the vehicle. The execution of the protocol ($\mathcal{P}$) is termed as an instance. For example, an instance of $\mathcal{P}$ run by $\mathcal{G}$ is denoted by $\mathcal{P}_{\mathcal{G}}^{(i)}$, where i is the i-th run of the protocol. A collection of protocol instances that are run by the same parties is called an oracle $\mathcal{O}_{\mathcal{G}}^{\mathcal{P}} = \{\mathcal{P}_{\mathcal{G}}^{(i)} | i = 1, 2, \ldots\}$. To prove the security of $\mathcal{P}$, we define a sequence of games. We start from an initial game, which reflects the real protocol. Then we modify it a few times until we end up with the final game that reflects an attack against an ideal protocol, where an adversary $\mathcal{A}$ has a negligible advantage of winning. As the advantage between any two consecutive games is negligible, we can conclude that $\mathcal{A}$ has a negligible advantage of winning the game against the real protocol.

Now a collection of queries that can be made by $\mathcal{A}$ are provided.

$-Send(\mathcal{P}_{\mathcal{G}}^{(i)}, M)$: This query models the adversarial ability of sending messages. $\mathcal{A}$ can initiate $\mathcal{P}$ by sending the query $Send(\mathcal{P}_{\mathcal{G}}^{(i)}, \text{“}start\text{”})$.

$-Hash(\mathcal{P}_{\mathcal{G}}^{(i)})$ & $-Enc(\mathcal{P}_{\mathcal{G}}^{(i)})$: These oracles model hash and encryption queries made by $\mathcal{A}$. The hash $\mathcal{H}$ and the encryption tables $\mathcal{E}$ are used to store entries of the hash and the cipher text for the inquired message (M) in the form (M, f) and (C, M). If M is stored already in $\mathcal{H}$ and $\mathcal{E}$, it will return the stored hash (f) and the stored ciphertext (C). Otherwise, it will return a random number (f') and (C') for the inquired M and stores the pair (M, f') and (M, C') in $\mathcal{H}$ and $\mathcal{E}$, respectively.

$-Session - key\ reveal(\mathcal{P}_{\mathcal{G}}^{(i)})$: This query models ability of $\mathcal{A}$ to force $\mathcal{P}_{\mathcal{G}}^{(i)}$ to reveal its session key. For instance, if $\mathcal{A}$ issues this query, then $\mathcal{P}_{\mathcal{G}}^{(i)}$ returns the ith session key.

$-Session - state\ reveal(\mathcal{P}_{\mathcal{G}}^{(i)})$: The ability of $\mathcal{A}$ in learning the internal states of $\mathcal{G}$ is modelled by this query. When $\mathcal{A}$ issues this query, $\mathcal{P}_{\mathcal{G}}^{(i)}$ returns secret random numbers of the ith session.

$-Corrupt(\mathcal{P}_{\mathcal{G}}^{(i)})$: $\mathcal{A}$ uses this query to acquire the long-term secret keys held by $\mathcal{G}$.

$-Execute(\mathcal{P}_{\mathcal{G}}^{(i)})$: This query models the passive attack by $\mathcal{A}$. The adversary reads all the messages exchanged between parties that run the instance $\mathcal{P}_{\mathcal{G}}^{(i)}$.

$-Remover(\mathcal{P}_{\mathcal{G}}^{(i)})$: This query models the expiry time of sessions. For instance if the query is issued to $\mathcal{P}_{\mathcal{G}}^{(i)}$, then the ith session time is expired for $\mathcal{G}$. We use this query to prove the PFS of the protocols.

$-Test(\mathcal{P}_{\mathcal{G}}^{(i)})$: This query can be asked by $\mathcal{A}$ once only at the very end of the game. The asked party $\mathcal{G}$, tosses an unbiased coin b. If $b = 1$, then it returns the actual session key. Otherwise, it returns a random number. $\mathcal{A}$ wins the game if it is able to identify b.

Remark 1. The hash $H(\cdot)$ and encryption $Enc(\cdot)$ oracles are instances of the random oracle (RO).

Now, we are ready to introduce notions and definitions needed in our analysis to make precise statements about security properties. We refer the definition of partnership in the work in [38] and session identification in the work [39].

Definition 1. (Freshness) The ith session is *fresh* if $\mathcal{P}_{\mathcal{G}}^{(i)}$ ends in the accepted state even when the following combination of queries are asked by $\mathcal{A}$ after the execution of $Test(\mathcal{P}_{\mathcal{G}}^{(i)})$: ($i$) $Session - state\ reveal(\mathcal{P}_{\mathcal{G}}^{(\neg i)})$, ($ii$) $Session - state\ reveal(\mathcal{P}_{\mathcal{G}}^{(\neg i)})$ and $Session - key\ reveal(\mathcal{P}_{\mathcal{G}}^{(\neg i)})$ and (iii) $Corrupt(\mathcal{P}_{RSU\ \&\ Vehicle}^{(i)})$ and

$Corrupt(\mathcal{P}^{(i)}_{Alice\ \&\ Bob})$ in the TV2I and KV2VP protocols, after the execution of $Remover(\mathcal{P}^{(i)}_{\mathcal{G}})$.

Definition 2. (Negligible function) A function $\mu(n)$ is *negligible*, if for all $c > 0$, there exists $n_0 \in \mathbb{Z}^+$ such that $|\mu(n)| \leq \frac{1}{n^c}$ for all $n \geq n_0$.

Definition 3. (ECDLP, ECDHP and ECq-SDHP assumption) Any probabilistic polynomial time algorithm $\mathcal{A}$ has a negligible advantage in solving ECDLP, ECDHP and ECq-SDHP. If n is the number of bits used to represent a member of the groups, then

$$Adv_{\mathcal{A}}^{ECDLP/ECDHP/ECq-SDHP}(n) \leq \mu(n).$$

Definition 4. (Semantic security) Let a protocol $\mathcal{P}$ with the security level n, an adversary $\mathcal{A}$ who interacts with the fresh session, and a query $Test(\mathcal{P}^{(i)}_{\mathcal{G}})$ with a random bit b which is asked by $\mathcal{A}$ in the session. Then the protocol $\mathcal{P}$ is *semantically secure* if the advantage of $\mathcal{A}$ is negligible or $Adv_{\mathcal{A}}^{SS}(\mathcal{P}, n) = Pr[\mathcal{A}(\mathcal{P}, n) = b] - \frac{1}{2} \leq \mu(n)$.

Difference lemma [40] Let W, X and Y be events defined in some probability distribution, and suppose that $W \wedge \neg Y \equiv X \wedge \neg Y$. Then, $Pr[W] - Pr[X] \leq Pr[Y]$.

Theorem 1. *Consider the TV2I protocol (a.k.a. $\mathcal{P}$) and the following assumptions: a long term secret key can be guessed with an advantage Adv_{sk}, ECDLP can be solved with an advantage Adv_{ECDLP}, the hash function generates MACs with ℓ bits, the encryption function generates cipher texts with k bits, the size of a dictionary is $\mathcal{D}$, a random number of length m bits, and $\mathcal{A}$ has access to all public oracles and can issue a polynomial number of queries. Then semantic security of $\mathcal{P}$ can be broken by $\mathcal{A}$ with the advantage $Adv_{\mathcal{P}}^{SS} \leq \frac{q^2_{hash}}{2^\ell} + \frac{q^2_{Enc}}{2^k} + \max\{\max(Adv_{sk}, \frac{1}{2^m}), (\frac{1}{2^m} + Adv_{sk}), (Adv_{ECDLP} + \frac{q^2_{Enc}}{2^k})\} + Adv_{sk} + Adv_{ECDLP} + \max\{\frac{1}{2^m}, \frac{q^2_{Enc}}{2^k}, (\frac{1}{2^m} + Adv_{sk} + Adv_{ECDLP})\}$, where q_{hash}, q_{enc} and q_s stands for the number of queries to the hash oracle, the encryption oracle and the send oracle, respectively.*

The proof of this theorem is provided in the appendix A.1.

Lemma 2. $e(Cert^*_{Alice}, PKV^*_{Alice} + t_d.Y) \iff e(P, Q)$.

The correctness proof of this lemma is presented in Appendix A.2.

Theorem 3. *Consider the KV2VP protocol (a.k.a. $\mathcal{P}$) and the following assumptions: a long term secret key can be guessed with an advantage Adv_{sk}; ECDLP, ECDHP and ECq-SDHP can be solved with an advantage Adv_{ECDLP}, Adv_{ECDHP}, $Adv_{ECq-SDHP}$, respectively; the hash function generates MACs with ℓ bits; the encryption function generates cipher texts with k bits and a random number of length m bits; and $\mathcal{A}$ has access to all public oracles and can issue a polynomial number of queries. Then semantic security of $\mathcal{P}$ can be broken by $\mathcal{A}$ with the advantage $Adv_{\mathcal{P}}^{SS} \leq \frac{q^2_{hash}}{2^\ell} + \frac{q^2_{Enc}}{2^k} + \max\{\max\{\frac{q^2_{hash}}{2^\ell}, \frac{q^2_{hash}}{2^\ell} + \frac{1}{2^m}, Adv_{ECDLP}\}, Adv_{ECq-SDHP}, Adv_{ECDHP}, \frac{q^2_{Enc}}{2^m}, \frac{1}{2^m} + Adv_{sk}\}\} + 3 \cdot Adv^2_{sk} + 2 \cdot Adv_{sk} + \max\{Adv_{ECDLP}, \max(\frac{q^2_{hash}}{2^\ell} + Adv_{ECDLP})\}$, where q_{hash} and q_{enc} stands for the number of queries to the hash and encryption oracles, respectively.*

The proof for this and the previous theorems are given in the appendix A.3.

6.2. Informal security analysis

Proposition 4. *The protocols TV2I and KV2VP provide session key freshness.*

In TV2I, the session key GK^* generated by KDC is fresh for every session because K is constructed using random λ, which is chosen for every session. Hence, the freshness of the session key is attained for every session in TV2I. In KV2VP, GK^* is verified for freshness based on ver_{gk} as ver_{gk} is incremented for every key update. Alice can check the validity of increment of ver_{gk} by computing $e(GK_{part1}, GK_{part2} \cdot Y) \iff e(P, Q)$. Therefore, the freshness of the session key is attained for every session in KV2VP.

Proposition 5. *For the protocols VUAP, TV2I and KV2VP, any replay of messages exchanged during a protocol execution will be identified by the parties who follow the protocols.*

VUAP –If $\mathcal{A}$ replays Step 3 in Section 5.5 to authenticate as a legitimate driver/owner, the vehicle will reject the message with non-negligible probability. The replay can be identified by the vehicle because the initial check is made for the time stamp based on the accepted network delay ΔT (Step 4 in Section 5.5).

TV2I – The adversary can replay the steps 3, 6, 11, 13, 16 and 18 in Section 5.6. If the adversary replays Step 3 and Step 6 in Section 5.6 to the RSU and KDC, the RSU may accept Step 3; however, KDC will reject the replay as SID between the vehicle and KDC are synchronised. And SID is updated in every session. If $\mathcal{A}$ replays Step 11 and Step 13 in Section 5.6, the RSU and the vehicle will reject the message with high probability because the RSU and the vehicle randomly choose R and S for every session, which is utilised to decrypt the ciphertexts ζ_R and ζ_S. If $\mathcal{A}$ replays Step 16 and Step 18 in Section 5.6, then the RSU may accept the message. But KDC will reject the message since synchronisation of SID^* is needed for the acceptance.

KV2VP – If $\mathcal{A}$ replays the steps 4, 9 and 14 in Section 5.7, then the parties will identify the replay as every message holds the time stamp (T_c) in it.

Proposition 6. *For the protocols VUAP, TV2I and KV2VP, impersonation attack against them fails with an overwhelming probability.*

VUAP – Since two secrets m and n are required in the protocol (refer Section 5.5), the adversary cannot succeed in the masquerading attack without knowing the secrets. Even if $\mathcal{A}$ guesses the password in polynomial time and constructs m, it cannot masquerade without knowing n.

TV2I – Without knowing the long-term keys (L_{VK} and L_{RSU_i}) in Section 5.6, $\mathcal{A}$ can not masquerade with high probability.

KV2VP – To mount a successful masquerading attack in KV2VP, $\mathcal{A}$ must know the long-term secrets (x_A and x_B) in Section 5.7.

Proposition 7. *Given the protocols VUAP, TV2I and KV2VP, the protocols are resistant against MITM attack.*

In the MITM attack, $\mathcal{A}$ replay and masquerade by recording any of the previous communication among the protocol parties. Since we argued that the protocols are resilient to the replay and masquerading attack, any attempt to successfully mount the MITM attack will not happen with high probability.

Proposition 8. *Given the KV2VP protocol, KDC can identify the misbehaving vehicles, and it can revoke misbehaving vehicles.*

Consider Alice in Section 5.7. The responsibility of Alice is to prove her validity (i.e. $t_d < t_c$) to Bob to get key. After the expiry of her departure time, if Alice illegitimately tries to acquire the group key, then it will be identified by Bob (Lemma 2 in Section 6.1). If Bob complains Alice to KDC using her credential received from Step 4 in Section 5.7, then KDC can identify and revoke Alice by identifying her identity from the public key and the certificate.

The role of Bob in the protocol is to deliver an updated key to Alice. If Bob misbehaves by injecting an invalid group key (i.e. contrary to the verification step of Step 13 in Section 5.7 of the protocol), then Alice can report the misbehaviour to KDC using the message from Step 9 in Section 5.7. Like Alice, Bob can be identified and revoked by KDC. Therefore, KDC can revoke both vehicles when they misbehave in the protocol.

Proposition 9. *For the VUAP protocol, online and offline password guessing attack succeeds with negligible probability.*

$\mathcal{A}$ can guess the password by subjecting either online or offline password guessing attack. But the need of n to compute y restricts the adversary in the successful mounting of the attack. As n is a hash value, which is computed using the secret of KDC δ, it cannot be obtained by $\mathcal{A}$ with non-negligible probability. Thus, online and offline password guessing attack against VUAP fails with non-negligible probability.

Proposition 10. *Given the VUAP protocol, lost smart card verifier attack fails with non-negligible probability.*

After attaining the smart-card, if the adversary tries to authenticate as a legitimate driver/owner, it cannot validate itself. The need for the password will restrict the adversary from authentication. This means that a valid y can be constructed using m and n. As the adversary does not know m, a valid y cannot be constructed. Hence, the protocol resists the lost smart-card verifier attack with non-negligible probability. It should be noted that this attack and the password guessing attack cannot be conjunctive.

Proposition 11. *Given the VUAP, TV2I, KV2VP protocols, any attempt of DoS can be detected by parties of the protocols.*

VUAP – If Step 3 in Section 5.5 is blocked, then the vehicle user can identify the DoS as the access to the vehicle is denied for the vehicle user.

TV2I – Out of the steps 3, 6, 11 and 13 in Section 5.6, if anyone of them is blocked, then the vehicle can identify the DoS in the channel because of unreachability of key from Step 13. If Steps 11, 13, 16 and 18 in Section 5.6 are blocked, then KDC can identify the DoS because of the requirement of the key confirmation step in Step 16. KV2VP – If Steps 4, 9 and 14 in Section 5.7 are blocked, Alice and Bob can know about DoS due to not getting a response as per the protocol.

7. Formal security verification

This section presents the formal verification of our protocol suite. Initially, the Tamarin tool and authentication properties are introduced. For brevity, we explain the description security protocol theory (SPTHY) format of TV2I alone. The entire set of scripts of the new protocol suite can be found in the link [41].

7.1. Tamarin overview

Tamarin is a "state-of-the-art" tool that is used for the verification of the security properties claimed by cryptographic protocols. The tool is highly versatile to check the security properties for different adversaries. This means that the tool can be deployed for testing protocols for the Dolev-Yao adversary [42] and user-specific adversarial capabilities. In addition, the tool possesses the following attractive features: supports modelling of global states in protocols [43], theorem proving of security goals can be accomplished using lemmata, relevant graphs illustrate discrepancy in security goals/claims in interactive mode. The complete description of the tool and its potentials can be found in the manual [44].

Protocols in Tamarin should be described in SPTHY format. It facilitates the description of protocols using states, facts and rules. The states are used to store variables and constants in protocols which denote knowledge of participants of a protocol. Facts are used to declare the nature of variables and channels. For instance, the Fact that denotes freshness is declared as `Fr()` and channels as `In()` and `Out()`. A rule is always ascertained of the format `[L]-[A]->[R]`, where `L`, `A` and `R` denote premise, action and conclusion, respectively. For every rule, the premise is the beginning state that is updated according to the action. The action can be several events in a protocol, such as a send and receive event, security properties and adversarial queries. The conclusion is the end state in a rule.

The predicates of the mutual entity authentication are as follows: aliveness, weak agreement, non-injective agreement and injective-agreement. The authentication claims follow a hierarchy. Injective agreement sits on the top of the authentication claims [45]. The injective agreement is guaranteed on the following conditions: a source and an intended destination agree on variables. For every committed variable by a source, there uniquely exist an intended destination that accepts the variable. The authentication properties can be verified using commit and running actions in the tool as parties of protocols agree on data items. In general, any properties supported by the tool can be verified using lemmata that are described using the first-order logic. Before verifying security properties, the reachability of states on the specification of a protocol is verified using a lemma exist-trace.

7.2. Description of SPTHY of VUAP

We perform formal verification of VUAP, and we verify the security properties governed by VUAP. We elaborate the SPTHY format of VUAP. The details of SPTHY format of VUAP could help the reader, understanding about the protocol description in Tamarin. Since the SPTHY script of VUAP [41] has more than 100 lines, the rules and the security properties in VUAP is discussed. The script starts with the header `theory VUAP`, the first line of the SPTHY script of VUAP. The second line `begin` indicates the start of the protocol description. The third line `builtins: hashing` presents the inbuilt cryptographic function, which is used in the protocol. The fourth line `functions:h1/1, h2/1` denotes the name of the public function that is utilised in the protocols. The below snippet presents the first four lines of the script.

```
theory VUAP
begin
builtins: hashing
functions: h1/1, h2/1
```

The next line `rule register_vehicle_credentials` specifies the registration of protocol party $Vehicle_i$ with KDC. The next few lines of the script are presented below. The symbols in the rule ˜, $ and ! specify the nature of the Fact, namely fresh values, public and persistent which means the assignment is constant throughout the completion of execution of a protocol, respectively. Refer line number 9 in the below snippet.

```
rule register_vehicle_credentials:
let
m=h(<~PWD>)
n=h(<m,~delta,~Vehicle_ID>)
y=h(<m,n>)
C=h(<y>)
in
[Fr(~PWD), Fr(~delta), Fr(~Vehicle_ID)]
                                  --[]->
[!LtVdK($Vehicle_driver,$KDC,<~PWD>),
!LTK($KDC,~delta)
,!LTKTR($KDC,$TR,C),!LTKSC($KDC,$SC,<n>)]
```

The rules `rule initialise_vehicle_driver`, `rule initialise_TR`, `rule initialise_SC` and `rule initialise_KDC` are described to initialise the state spaces of the vehicle driver, tamper-resistant module, smart-card and KDC with the long-term keys and roles. The state spaces are initialised using the action in the rule. For example, the action at TR contains `Create(TR, ĩd), Role('TR')`. Create a role TR with fresh `id` such that every execution/session made by TR will be fresh (refer line number 4 below).

```
rule initialise_TR:
[Fr(~id),!LTKTR(KDC,TR,C),
!LtVdK(Vehicle_driver,KDC,<PWD>)
,!LTKSC(KDC,SC,<n>)]
--[Create(TR, ~id), Role('TR') ]->
[St_TR_1(TR, ~id,C, KDC,Vehicle_driver,SC)]
```

The next rule `rule send_1` presents the first communication of $VehicleUser$, i.e, Step 3 in VUAP (ref Section 5.5). A snippet of the rule *send*_1 is presented below. The premise of the rule contains the initialised state of $VehicleUser$ (line 8 in the below snippet), timestamp `T_c` and password shared between $VehicleUser$ and KDC (line 7 in the below snippet). The action of the rule contains the following events: (i) $Send(< Vehicle\_driver, < Tc, M1 >>)$ specifying the first communication (line 10 in the below snippet), (ii) $Role('Vehicle\_driver')$ specifying the role of $VehicleUser$ (line 11 in the below snippet), (iii) $Secret(< Vehicle\_driver, SC, PWD, n >)$ specifying the secret involved in the protocol (line 12 in the below snippet) and (iv) $Commit\_strongA(PWD)$ specifying the agreement on password (line 12 in the below snippet).

```
rule send_1:
let
m=h(<PWD>)
y=h(<m,n>)
M1=h(<h(<y>),~Tc>)
in

 [Fr(~Tc),!LtVdK(Vehicle_driver,KDC,<PWD>),
St_Veh_dri_1(Vehicle_driver, ~id,PWD, KDC,
  TR,SC),
 St_SC_1(SC, ~id,n, KDC,Vehicle_driver,TR)]
--[Send(<Vehicle_driver,<~Tc,M1>>),
Role('Vehicle_driver'),
Secret(<Vehicle_driver, SC, PWD,n >),
 Commit_strongA(PWD)]->
 [St_Veh_dri_2(Vehicle_driver, ~id,PWD,~Tc,
  KDC,TR,SC),
 St_SC_2(SC, ~id,n,~Tc,M1, KDC,
 Vehicle_driver,TR),Out(<~Tc,M1>)]
```

The final rule `rule Recv_1` presents the received communication of $Vehicle$, i.e., after Step 3 in VUAP (ref Section 5.4). The list of actions is specified in line 9 and line 10 of the below snippet. Finally, the description ends with the syntax end after the declaration of security properties that is explained in the following section.

```
rule Recv_1:
let
m=h(<PWD>)
y=h(<m,n>)
M1=h(<h(<y>),Tc>)
in
[In(<Tc,M1>),!LTKTR(KDC,TR,C),
!LTKSC(KDC,SC,<n>),
St_TR_2(TR, ~id,C,PWD, KDC,
Vehicle_driver,SC)]
--[Receive(<TR,<n>>), Role('TR'),
Check(<h(C,Tc),M1>),Running_strongA(PWD),
 Unique(M1) ]->
[St_TR_3(TR, ~id,C,PWD,Tc,M1, KDC,
Vehicle_driver,SC),
St_Veh_dri_3(Vehicle_driver, ~id,PWD,Tc,M1,
KDC,TR,SC)]
```

7.3. Verification of VUAP

Security properties are verified by specifying it in the first-order logic. VUAP has the following lemmata: 1) `executable`, 2) `lemma Injectiveagreement_Vehicle` and 3) `secrecy`.

1) `lemma executable: all-traces "All Vehicle_driver m #i.`
`Send(<Vehicle_driver,<m»)@i ==> Ex TR m #j.`
`Receive(<TR,<m») @ j "`

The lemma executable states that for all `Vehicle_driver` that sends message `m` at time `i`, there exists `TR` that receives the `m` at time `j`. This lemma is essential to verify the reachability of states.

2) `lemma Injectiveagreement_Vehicle: " (All PWD #i . Commit_strongA(PWD)@ #i ==> (Ex #j . Running_strongA(PWD) @ #j & #j < #i) & (not (Ex #j . Commit_strongA(PWD) @ #j &`
`not (#i = #j))) )"`

The above lemmata verify the injective agreement property. The lemma verifies that for each commit by a party there uniquely exists a running partner. The lemmata state that for all committed value `PWD` at time `i`, there exist running a part running `PWD` at `j`, where `j` is precedent to `i`, and it can not be that there exist a committed value `PWD` at `j`, and `j` and `i` are equal valued. It is noted that the injective agreement alone is verified as it sits on the top of the hierarchy of authentication claims.

3) `lemma secrecy: all-traces "All Vehicle_driver SC PWD n #i.`
`Secret(<Vehicle_driver, SC, PWD,n >) @ i ==>`
`not( Ex PWD n #r.K(<PWD,n>)@r)"`

The above lemma states that for every `Vehicle_driver`, smart card `SC`, `PWD` and `n`, password (PWD) and n are secret if those values does not exist in the knowledge of the adversary. This lemma specifies the secrecy of the password and the smart-card secret n.

4) `lemma Anonymity: exists-trace "not(Ex Vehicle_driver M1 #m.`
`K(<Vehicle_driver,M1>) @ m)"`

Table 3
Verified lemmata of the new protocol suite.

Lemma / Protocol	Injective agreement	Session key/secret security	KKS	PFS	Conditional anonymity	Unlinkability
VUAP	✓	✓	NA	NA	✓	✓
TV2I	✓	✓	✓	✓	✓	✓
V2V	✓	✓	✓	✓	✓	✓

This lemma states that it can not be that there exist a vehicle driver and his/her interaction ($M1= H(H(y) \parallel T_c)$) in the knowledge of the adversary.

5) `lemma Unlinkability: all-traces "All #i #m M1 Tc. Unique(M1) @ i & Unique(M1) @ m & K(<Tc>) @ i & K(<Tc>) @ m==> #i=#m"`

This lemma states that in any two communications involving vehicle driver (i and m) in which the adversary knows the temporal information (Tc) of the communications, the message sent by the driver is uniquely generated.

Mapping security goals: The verified security goals of the VUAP protocol are mutual entity authentication, the secrecy of the password and smart-card credential, the resistance of the protocol towards the following attacks: replay, masquerading and MITM and the privacy properties, namely conditional anonymity and unlinkability. The first lemma proves the reachability of states in the protocol. Verification of the second lemma achieves the mutual authentication and resistance of the protocol towards the attacks. Verification of the third lemma guarantees the security of the password and the smart-card credential. The fourth and fifth lemmata prove the privacy properties of the protocol. Table 3 presents the verified lemmata of VUAP, TV2I and KV2VP. In the below table, NA denotes not applicable.

8. Performance comparison

This section compares the computation efficiency, security and functional attributes of our protocol suite with other protocols. We use the TEPLA library [46] to evaluate the various elliptic curve operations. BN-254 curve with the base field size of 512 bits is chosen for the evaluation. For the details of the curve, reader is requested to refer Section 3.2. We run the evaluation for several times, and we consider the average computation time as the computation times of the primitives in the library. We use the pycrypto library [47] for evaluating the execution time of hash and encryption algorithms. The computation time of a hash operation is evaluated by running SHA-256 algorithm, and the computation time of encryption operation is evaluated by running AES-128 algorithm. For the AES evaluation, plain text size of 16 bytes, initialisation vector of 16 bytes and the key size of 128 bits is chosen with CBC mode operation. For the hash function evaluation, plaintext size of 32 bytes and key of 128 bits are selected for the evaluation. All the operations and algorithms are ran for 1000 iterations, and the average value is taken as the computation time (C_T) of the operations and algorithms. CPU cycles per operation (CPO) is derived accordingly.

The evaluation is done on two platforms with the following specifications: Ubuntu (V 16.04) with 64 bit Intel core I7 central processing unit (CPU) at 2.6 GHz operating frequency and Raspian operating system (OS) with 32 bit ARM CPU at 900 MHz operating frequency. The average computation time and CPO of the operations and algorithms on the Ubuntu and the Raspian OSs are presented in Tables 4 and 5, respectively. For convenience, we define the following notations. GT_A is the time needed for a point addition on the curve, GT_M is the time required for a point multiplication on the curve, GT_P is the time required for a pairing operation on the curve, GT_E is the time needed for an exponentiation operation on the curve, T_E is the time needed to perform an encryption/decryption operation, T_H is the time needed for performing a hash operation and T_{XOR} is the time intended for performing a XOR operation. The execution time of the XOR operation is not considered as it is negligible compared to the execution time of other operations. It is noted that the time duration for exponentiation operation (GT_E) and the point multiplication are approximately equal, i.e. $GT_M \approx GT_E$ [6].

We implement the tate pairing on a supersingular elliptic curve $E: y^2 = x^3 - x + 1$ in the affine coordinates defined over a Galois Field $GF(3^m)$ [48]. The following environment is chosen for the evaluation of the ate pairing in BN curve: Ubuntu (V 16.04) with 64 bit Intel core I7 central processing unit (CPU) at 2.6 GHz operating frequency. We estimate the computation time of a point multiplication, point addition and pairing operations as 1.165 s, 0.1337 s and 8.125 s, respectively. The simulation is iterated over several times, and the average value is taken as the computation time of various operations. We confirm that the execution time of the ate pairing is higher than the tate pairing.

8.1. Computation efficiency

The computation efficiency of our protocols is presented by comparing with the related protocols regarding a total number of the operations and algorithms (TOA), the time duration for computations of TOA and CPU cycles. C_T and CPO are taken from Table 5 to present the computation efficiency of the VUAP protocol.

VUAP: VUAP is used for driver/owner authentication. VUAP requires $3T_H$ ($\approx$ 474.21 μs), and LGKP need $3T_H + T_{XOR}$ ($\approx$ 474.21 μs) TOA. 2FLIP needs $9T_H + 5T_{XOR}$ TOA ($\approx 14.22 \times 10^{-4}$ s). But ABAKA requires unit computation since password and identity verification are done without any of the mentioned operations and algorithms. VUAP and LGKP require 427.17×10^3 CPU cycles, whereas 2FLIP needs $\approx 1281.51 \times 10^3$ CPU cycles.

TV2I: TV2I is used for distributing keys to vehicles from KDC. For the computations involving vehicles, Table 5 is used, and for the computations involving RSUs and KDC/trusted authority, Table 4 is used. Table 6 presents the computation efficiency of TV2I with related protocols. For comparison, we consider the construction time of the involved primitives. The time duration for ordinary signature computation is calculated as a sum of a hash and exponentiation operations. For signatures involving elliptic curve system, the signature computation time is calculated as a sum of a hash and point multiplication operations. Though FKDP and 2FLIP is computationally more efficient than TV2I, both of them do not satisfy all security and functional attribute (refer Table 8).

KV2VP: KV2VP is used for exchanging keys between vehicles. Tables 4 and 5 are utilised to present the computation efficiency of KV2VP and V2V of FKDP. KV2VP requires $5GT_M + 3T_H + 2GT_P + T_E$ TOA, and V2V of FKDP needs $\approx 7GT_E + 3GT_P + 5T_E$ TOA. While using Table 4 V2V of FKDP need the time durion of 34.17 ms and 51.06 ms, correspondingly. KV2VP requires the CPU cycles of 102.52×10^6, and V2V of FKDP needs 153.24×10^6. When Table 5 is utilised, KV2VP is in need of 335.63 ms, and V2V of FKDP require the time duration of 511.31 ms. When Table 5 is used, KV2VP require 307.03×10^6 CPU cycles, and V2V of FKDP require the CPU cycles of 460.5×10^6.

PEM: The PEM is used for exchanging messages in V2I and V2V communications. The PEM protocol demands 2 T_H TOA that requires time of 316.14 μs and 0.28×10^6 CPU cycles. Azees et al.

Table 4
Average computation time and CPO of the cryptographic operations and algorithms on Ubuntu OS.

Notations	GT_A	GT_M	GT_P	T_E	T_H
C_T	0.2 μs	1.38 ms	13.45 ms	0.21 ms	55.07 μs
CPO	5.18×10^4	3.6×10^6	41.77×10^6	54.68×10^4	14.49×10^4

Table 5
Average computation time and CPO of the cryptographic operations and algorithms on Raspian OS.

Notations	GT_A	GT_M	GT_P	T_E	T_H
C_T	96.6 μs	36.28 ms	76.8 ms	5.39 ms	158.07 μs
CPO	86.48×10^3	32.68×10^6	69.18×10^6	4.85×10^6	142.39×10^3

Table 6
Computation efficiency of TV2I with related protocols.

Scheme/Protocol	TOA	Time	CPU cycles
TV2I	$10T_H + 12T_{XOR} + 1T_E$	≈1.17 ms	$\approx 19.8 \times 10^5$
LGKP	$6T_H + 2T_{XOR} + 1T_E + 2GT_E$	≈38.61 ms	$\approx 37.68 \times 10^6$
PAFH	$2T_H + 6T_{XOR} + 5GT_M$	≈111.71 ms	$\approx 105.52 \times 10^6$
LIAP	$T_H + 3T_{XOR} + 5GT_M$	≈41.95 ms	$\approx 47.22 \times 10^6$
2FLIP	$T_E + GT_M + T_H$	≈1.74 ms	$\approx 4.28 \times 10^6$
ABAKA	$T_H + 7GT_M + 2GT_A$	≈114.66 ms	$\approx 112.86 \times 10^6$
FKDP	GT_M	≈1.38 ms	$\approx 3.6 \times 10^6$

Table 7
Security and functional attributes of VUAP.

Attributes	VUAP	Ying et al. [15]	Wang et al. [11]	Huang et al. [12]
Conditional anonymity	✓	✗	✓	✗
Unlinkability	✓	✗	✓	✗
Authentication by the Vehicle	✓	✓	✓	✓
Replay attack	✓	✗	✓	✗
Masquerading attack	✓	✓	✓	✓
Password change phase	✓	✓	✓	✗
Password guessing attack	✓	✓	✓	✗
Lost smart card verifier attack	✓	✗	NA	NA

[6] protocol needs $T_H + 2GT_M$ TOA that require time duration of 72.71 ms and 65.50×10^6 CPU cycles. It is apparent that EAAP has enormously higher CPU cycles than the PEM protocol due to asymmetric key cryptography operations. The heavier computation restricts the serving capacity of RSUs. Figs. 4-A and 4-B present the serving capacity of an RSU of EAAP and the PEM protocol when the density of vehicles and their speed vary. The serving capacity is computed using the formula mentioned in [6], which is $C_{RSU} = \frac{p \cdot T \cdot r}{s \cdot d}$. C_{RSU}, p, T, r, s and d denote the serving capacity, the probability of sending the LBSI message by the RSU, TOA, coverage of the RSU, the speed of vehicles and density of vehicles, respectively. Here, the following values are used: $p = 0.5$, $T = 316.14$ μs for PEM, $T = 72.71$ ms for EAAP, $r = 300$ m, $5 \leq s \leq 10$ m/s and $200 \leq d \leq 400$. From Figs. 4-A and 4-B, it is conspicuous that the serving capacity can be increased dramatically for the PEM protocol in comparison with EAAP as the negative order of the serving capacity is increased by four times in the PEM protocol. The increase in the serving capacity of the PEM protocol is due to the utilization of symmetric crypto operations. Thus, PEM is computation efficient than its counterpart.

8.2. Security and functional attributes

Table 7, 8 and 9 show the comparison of security goals and functional attributes of VUAP, TV2I and KV2VP, respectively. In Tables 7, 8 and 9, the following notations are used: NA- not applicable, ✓- satisfies the attribute and ✗- does not satisfy the attribute.

From Table 7, it is apparent that 2FLIP satisfies most of the attributes in comparison with VUAP, but 2FLIP requires 50% more CPU cycles than VUAP regarding computation efficiency. Although LGKP and ABAKA have lesser computation efficiency than VUAP, they do not satisfy the privacy properties (ref Table 7). Hence, the VUAP protocol is computation efficient and robust regarding security among the protocols that preserve the privacy of vehicles.

From Table 8, the computation efficiency of TV2I is lesser than FKDP and equal to the protocol of 2FLIP. But TV2I is more robust regarding security and functionality features in comparison with its counterpart. The protocol does not depend on a mobility prediction model as in [23] and [22] because the protocol supports handoff capability for a vehicle with two RSUs ($(i-1)$th and $(i+1)$th) when moving away from the ith RSU. The protocol can assure detectability to the DoS attack as it includes the key confirmation step. The protocol has ≈67% lesser CPU cycles than PAFH and ≈ 70% than LIAP. Hence, among the protocols that support handoff capability, TV2I is more computation efficient and robust, and it supports more functional attributes. PAFH and LIAP demand a mobility prediction model, whereas TV2I does not require a mobility prediction model because the protocol supports the handoff capability. The protocol can assure detectability to the DoS attack as it includes the key authentication step.

Table 8
Security and functional attributes of TV2I with the related protocols.

Attributes	TV2I	Ying et al. [15]	Zhou et al. [23]	Li et al. [22]	Wang et al. [11]	Huang et al. [12]	Park et al. [17]
Conditional anonymity	✓	✓	✓	✓	✓	✓	✗
Unlinkability	✓	✓	✓	✗	✓	✓	✗
Mutual entity authentication (Vehicle and KDC^*)	✓	✓	✓	✓	✗	✓	✗
Mutual entity authentication (RSU and KDC^*)	✓	✓	✓	✓	✗	✗	✗
Explicit key authentication	✓	✗	✓	✓	✗	✗	✗
Key freshness	✓	✓	✗	✗	✓	✓	✓
Key integrity	✓	✗	NA	NA	✓	NA	✓
Forward secrecy	✓	✓	✗	✗	✓	✓	✓
Known-key secrecy	✓	✗	✗	✗	✗	✓	✓
Replay attack	✓	✓	✓	✓	✗	✓	✗
Masquerading attack	✓	✗	✓	✗	✓	✓	✗
MITM attack	✓	✓	✓	✓	✓	✓	✗
Dynamic RSU addition	✓	✓	✓	✓	✓	✓	✓
Handoff support	✓	✗	✓	✓	✗	✗	✗
LBSI service by RSU	✓	✗	✗	✗	✗	✗	✓
Independence of a movement prediction model	✓	NA	✗	✗	NA	NA	NA
Detectability to DoS	✓	✗	✗	✗	✗	✗	✗

Table 9
Security and functional attributes of KV2VP with related protocol.

Attributes	KV2VP	Park et al. [17]
Conditional anonymity	✓	✓
Unlinkability	✓	✓
Mutual entity authentication (Alice vehicle and Bob vehicle)	✓	✗
Explicit key authentication	✓	✗
Key freshness	✓	✓
Key integrity	✓	✓
Forward secrecy	✓	✓
Known-key secrecy	✓	✗
Replay attack	✓	✓
Masquerading attack	✓	✗
MITM attack	✓	✓
Detectability to DoS	✓	✗

From Table 9, KV2VP is more robust in security than FKDP. KV2VP is computation efficient than its counterpart. In a nutshell, among the protocols that support handoff authentication and areas not covered by V2I, the proposed protocols are computation efficient and robust in terms of security and offer more functional attributes.

9. Simulation performance

We evaluate the practicality of the proposed protocol suite and FKDP, regarding average key dissemination time (AKDT). We compare our protocols with FKDP because FKDP achieves faster average key dissemination time than other protocols. FKDP attains the amplified key dispatching due to the dissemination of keys made by the V2I and V2V communication modes, whereas the remaining protocols utilise the V2I communication mode alone for the key dissemination.

The simulation is done in Omnet++ network simulator with the integration of SUMO urban mobility simulator. For the simulation, ten lanes of 5 km $\times$ 1 km area are considered. To facilitate handoff capability, RSUs are made to transmit and receive messages continuously. The simulation is carried out for 100 vehicles and 1000 vehicles scenarios. Each scenario is divided into two cases depending on the speed of the vehicles. The two cases are for 30 km/hr and 70 km/hr. Each of the two cases is further divided into three cases depending on the coverage of RSUs: 10%, 30% and 50%. In particular, cases a, b, c, and d specify 100 vehicles at 30 km/hr, 100 vehicles at 70 km/hr, 1000 vehicles at 30 km/hr and 1000 vehicles at 70 km/hr, respectively. The simulations are iterated several times, and the average value of the key dissemination is taken for each case. Figs. 5-A present AKT for 100 vehicles, and 5-B present AKT for 1000 vehicles.

In general, the proposed protocols show similar performance as FKDP with slight improvements (i.e. reduced AKT) such as: (i) at 10%, 30% and 50% coverage for 30 km/hr scenario with 100 vehicles, and (ii) at 10% and 30% coverage for 70 km/hr scenario with 1000 vehicles. Overall, the simulation performance of our protocol suite is as fast as FKDP. Hence, our protocol suite is also a fast key dissemination protocol suite. As our protocol suite is a fast key dissemination protocol suite, it can reduce delay occurred for key dissemination.

10. Conclusion

In this paper, we analysed LGKP and FKDP; we demonstrated insecurities LGKP and FKDP. As a counter measure, we proposed

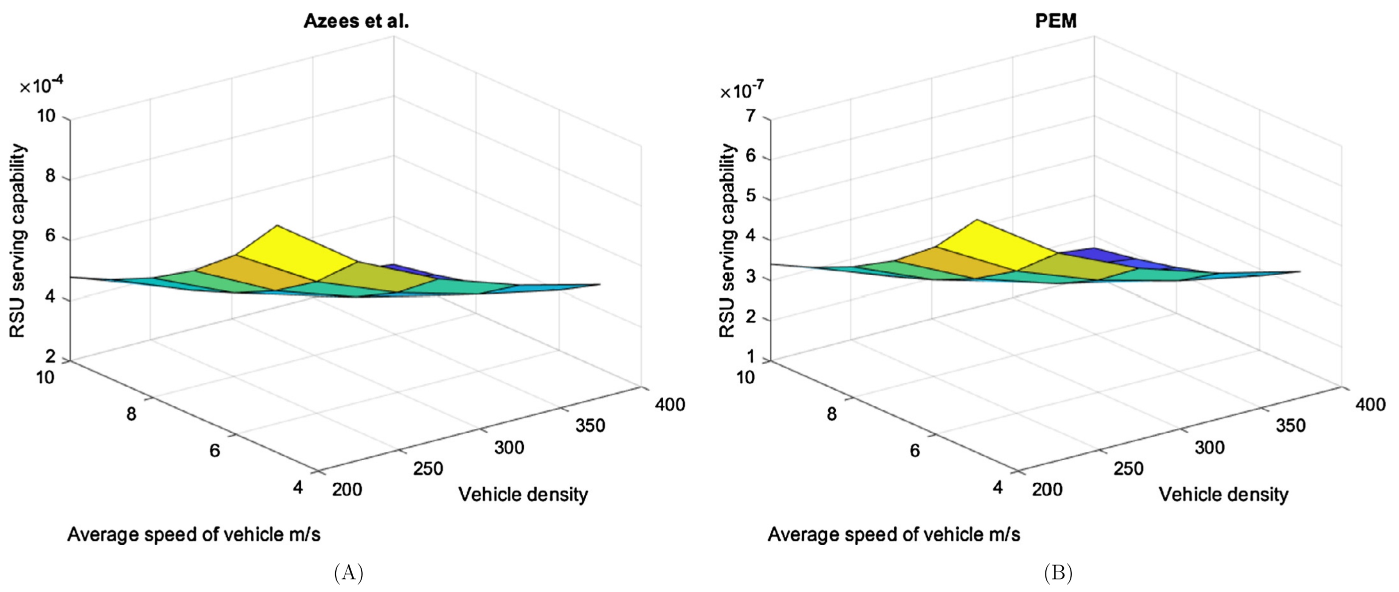

Fig. 4. (A) Protocol for Exchanging Messages (PEM) of Azees et al. (B) PEM of this work.

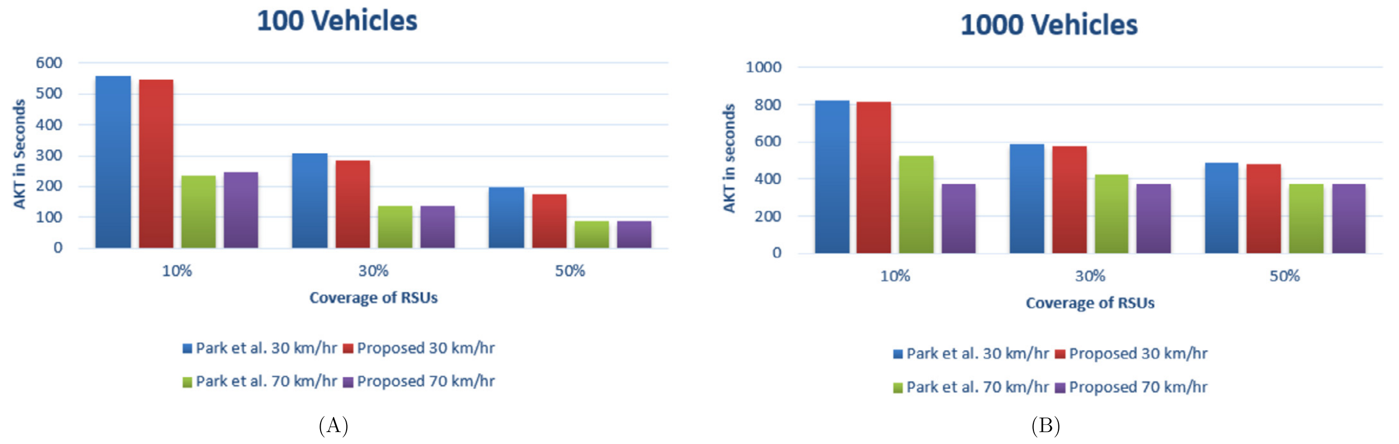

Fig. 5. Average key dissemination time (AKT) for (A) 100 and (B) 1000 vehicles.

a new protocol suite. The proposed suite had ten protocols. The new protocol suite assures several functional attributes. Specifically, TV2I ensures hand off capability and detectability to DOS and KV2VP offers the capability to check the time bounded validity of certificates of vehicles. The new protocol was computational efficient. The properties attained by the protocols were desirable for VANET. We proved the security of TV2I and KV2VP in the random oracle model. We performed an informal analysis of the protocols against various attacks and security goals. We presented a formal verification of VUAP, TV2I and KV2VP; the simulation performance of the protocols regarding average key dissemination time showed that the new protocol suite was also a fast key dissemination protocol.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We thank the anonymous reviewers and the associate editor for giving valuable comments and suggestions in order to improve the quality of this manuscript. Baker Palaniswamy has been supported by QUTPRA. Josef Pieprzyk has been supported by the Australian ARC grant DP180102199 and the Polish NCN grant 2018/31/B/ST6/03003.

Appendix A. Security Proofs

A.1. Proof of TV2I (Theorem 1)

We defined games G_i, and S_i is the event that $\mathcal{A}$ wins the game G_i, where $0 \le i \le 4$.

Game G_0: The game specifies the real-world attack against semantic security of $\mathcal{P}$. After collecting a polynomial number of queries to $Execute(\mathcal{P})$, $Enc(\mathcal{P})$ and $Hash(\mathcal{P})$, the adversary issues a $Test(\mathcal{P})$ query to an arbitrary session (s.t. ith session), which is in the accept state and the freshness holds. $\mathcal{A}$ wins the game if it can guess the bit b correctly. According to the definition of semantic security, $\mathcal{A}$ wins with an advantage

$$Adv_{\mathcal{P}}^{SS} = |Pr(S_0) - \frac{1}{2}|.$$

Game G_1: This game proceeds identically to the previous game unless there is a collision for MACs, i.e. $\sigma_{1..5}$. If the collision happens for σ_1, σ_2, $\sigma_3^{/*/**}$, σ_4 and σ_5, $\mathcal{A}$ can fake KDC as the registered vehicle, KDC as the legitimate RSU_i, the RSU for the cipher text (ζ_R), KDC for the key confirmation and KDC for the legitimate RSU, respectively. According to the birthday paradox, the number of queries to the hash table ($\mathcal{H}$) needs to be at least $\frac{q_{hash}^2}{2^\ell}$. We use the difference lemma to show that

$$|\ Pr[S_1] - Pr[S_0]\ | \leq \frac{q_{hash}^2}{2^\ell}.$$

Game G_2**:** This game is similar to the previous game unless there is a collision for the cipher texts (ζ_S & ζ_R). If the collision happens for ζ_S & ζ_R in the $\mathcal{E}$, $\mathcal{A}$ can force the vehicle and the RSU to decrypt ζ_S & ζ_R, respectively. According to the birthday paradox, the number of queries to the $\mathcal{E}$ must be at least $\frac{q_{Enc}^2}{2^k}$. According to the difference lemma

$$|\ Pr[S_2] - Pr[S_1]\ | \leq \frac{q_{Enc}^2}{2^k}.$$

Game G_3**:** This game corresponds to the KKS of $\mathcal{P}$, and it proceeds similarly to the previous game unless $\mathcal{A}$ identifies the current session key, GK^*. In this game, $\mathcal{A}$ asks $Session-state\ reveal(\mathcal{P}^{(\neg i)}_{Vehicle,RSU_i})$ and $Session-key\ reveal(\mathcal{P}^{(\neg i)}_{Vehicle,RSU_i})$ without losing the freshness of the session. That is, the queries are issued to the remaining sessions except the ith session to acquire session-specific random values (S, R and r) and session key (GK). Despite possessing the remaining session secrets and session keys, $\mathcal{A}$ requires either S or R of the ith session to know the ith session key. Otherwise, $\mathcal{A}$ needs the long-term secret ϵ and K_c, and the session-specific secret λ to construct the key. The advantage of $\mathcal{A}$ in attaining either S or R is $\max(Adv_{sk}, \frac{1}{2^m})$ since both the values can be identified by knowing the long-term secrets (L_{VK} and L_{RSU}) or replacing S and R by a random number, which is drawn randomly from an uniform distribution. The advantage of $\mathcal{A}$ for attaining the λ, K_c and ϵ are $\frac{1}{2^m}$, Adv_{sk} and Adv_{sk}, respectively. Alternatively, $\mathcal{A}$ can acquire ϵ by feeding $Y = \epsilon \cdot P$ to an ECDLP solver. The solver outputs ϵ with an advantage of Adv_{ECDLP}. Similarly, K can be acquired by querying $\mathcal{E}$ with the advantage of $\frac{q_{enc}^2}{2^l}$. The total advantage of $\mathcal{A}$ in this game is the maximum probability in the following advantages:

$$|\ Pr[S_3] - Pr[S_2]\ | \leq \max\{\max(Adv_{sk}, \frac{1}{2^m}), (\frac{1}{2^m} + Adv_{sk}),$$
$$(Adv_{ECDLP} + \frac{q_{Enc}^2}{2^k})\}.$$

Game G_4**:** This game presents the anonymity and the unlinkability of the $\mathcal{P}$, which differs by the fact that $\mathcal{A}$ traces the vehicle and links its communication. In this game, $\mathcal{A}$ asks $Session-state\ reveal(\mathcal{P}^{(\neg i)}_{Vehicle,RSU})$ without losing the freshness of the session to acquire the session-specific random values except the ith session values. That is, $\mathcal{A}$ issues the query to acquire r and λ of the remaining sessions. Although $\mathcal{A}$ holds r of the remaining sessions, it requires GK of the $(i-1)$th session to compute SID and link its communication. Given the $Session-state\ reveal(\mathcal{P}^{(\neg i)}_{Vehicle,RSU})$ query, the remaining session keys can be computed by knowing K_c and ϵ. K_c can be replaced with the advantage Adv_{sk}, and ϵ can be replaced in two ways: with the advantage Adv_{sk} and Adv_{ECDLP}. The advantage of $\mathcal{A}$ in this game according to the difference lemma is

$$|\ Pr[S_4] - Pr[S_3]\ | \leq Adv_{sk} + Adv_{ECDLP}.$$

Game G_5**:** This game presents the PFS of $\mathcal{P}$ and differs from the previous game if $\mathcal{A}$ identifies GK^* of the ith session. $\mathcal{A}$ asks $Corrupt(\mathcal{P}^{(i)}_{Vehicle})$ and $Corrupt(\mathcal{P}^{(i)}_{RSU_{(i-1),i,(i+1)}})$ to acquire the long-term secrects of the vehicle and the RSUs (L_{VK} and $L_{RSU_{(i-1),i,(i+1)}}$, respectively). As the freshness should not be violated, the queries are asked after the expiry of the session at the vehicle and the RSUs. The expiry of the session is achieved by executing the remover query $Remover(\mathcal{P}^{(i)}_{Vehicle,RSU_{(i-1),i,(i+1)}})$ at the vehicle and the RSUs. $\mathcal{A}$ needs either S or R, or λ of the ith session and ϵ and K_c to acquire and compute the key, respectively. $\mathcal{A}$ can replace either S or R by a random number with advantage $\frac{1}{2^m}$. Alternatively, $\mathcal{A}$ can query $\mathcal{E}$ to acquire the key with advantage $\frac{q_{Enc}^2}{2^k}$. To compute the key, $\mathcal{A}$ can replace λ by a random number chosen from a uniform distribution with advantage $\frac{1}{2^m}$. ϵ and K_c can be acquired with the advantage Adv_{ECDLP} and Adv_{sk}. The advantage of $\mathcal{A}$ in this game is the maximum probability in one of the three advantages: $\frac{1}{2^m}$, $\frac{q_{Enc}^2}{2^k}$ and $\frac{1}{2^m} + Adv_{sk} + Adv_{ECDLP}$. Applying the difference lemma, we get

$$|\ Pr[S_5] - Pr[S_4]\ | \leq \max\{\frac{1}{2^m}, \frac{q_{Enc}^2}{2^k}, (\frac{1}{2^m} + Adv_{sk} + Adv_{ECDLP})\}.$$

The game G_5 is the final one, where the probability of $\mathcal{A}$ in winning the event S_5 is,

$$Pr[S_5] = \frac{1}{2}$$

By the triangular inequality, we have:

$$\begin{aligned}
&|\ Pr[S_0] - Pr[S_5]\ | = |\ Pr[S_0] - Pr[S_1] + Pr[S_1] - Pr[S_2] \\
&+ Pr[S_2] - Pr[S_3] + Pr[S_3] - Pr[S_4] \\
&+ Pr[S_4] - Pr[S_5]\ | \\
&\leq |\ Pr[S_0] - Pr[S_1]\ | + |\ Pr[S_1] - Pr[S_2]\ | + \\
&|\ Pr[S_2] - Pr[S_3]\ | + |\ Pr[S_3] - Pr[S_4]\ | + \\
&|\ Pr[S_4] - Pr[S_5]\ |.
\end{aligned}$$

This means that

$$|\ Pr[S_0] - Pr[S_5]\ | = |\ Pr[S_0] - \frac{1}{2}\ |.$$

$$\begin{aligned}
Adv_{\mathcal{P}}^{SS} \leq\ & \frac{q_{hash}^2}{2^\ell} + \frac{q_{Enc}^2}{2^k} + \\
& \max\{\max(Adv_{sk}, \frac{1}{2^m}), (\frac{1}{2^m} + Adv_{sk}), \\
& (Adv_{ECDLP} + \frac{q_{Enc}^2}{2^k})\} + Adv_{sk} + Adv_{ECDLP} + \\
& \max\{\frac{1}{2^m}, \frac{q_{Enc}^2}{2^k}, (\frac{1}{2^m} + Adv_{sk} + Adv_{ECDLP})\}.
\end{aligned}$$

This concludes our proof. □

A.2. Correctness Proof of KV2VP

We present the correctness proof of Step 6 and Step 11 in KV2VP. The step 6 is presented in the lemma below.

Proof of Step 6 (Lemma 2). $e(Cert^*_{Alice}, PKV^*_{Alice} + t_d.Y) \iff e(P, Q)$. It follows:

$$\begin{aligned}
& e(Cert^*_{Alice}, PKV^*_{Alice} + t_d \cdot Y) \\
\iff & e(\frac{1}{H(x^*_A \parallel \delta \parallel ID)^* + \delta t_d} \cdot P, H(x^*_A \parallel \delta \parallel ID)^* \cdot Q + t_d \cdot \delta Q)
\end{aligned}$$

$$\iff e(\frac{1}{H(x_A^* \parallel \delta \parallel ID)^* + \delta t_d}.P, (H(x_A^* \parallel \delta \parallel ID)^* + t_d.\delta).Q)$$

$$\iff e(P, Q)^{\cancel{\frac{H(x_A^* \parallel \delta \parallel ID)^* + \delta t_d}{H(x_A^* \parallel \delta \parallel ID)^* + \delta t_d}}}$$

$$\iff e(P, Q).$$

Hence the proof. □

A.3. Proof of KV2VP

Proof of Theorem 3. We defined games G_i, such that $i \in [0, 5]$. S_i is the event that $\mathcal{A}$ wins the game G_i, where $i = 0, 1, 2, 3, 4, 5$.

Game G_0: The actual attack against the semantic security of $\mathcal{P}$ is presented in this game. After collecting a polynomial number of queries to $Execute(\mathcal{P})$, $Enc(\mathcal{P})$ and $Hash(\mathcal{P})$, the adversary issues a $Test(\mathcal{P})$ query to an arbitrary session which is in the accept state and the freshness holds. $\mathcal{A}$ wins the game if it can guess the bit b accurately. According to the definition of semantic security, $\mathcal{A}$ wins with an advantage

$$Adv_{\mathcal{P}}^{SS} = |Pr(S_0) - \frac{1}{2}|.$$

Game G_1: This game is similar to the previous game unless there is a collision for MACs, i.e. σ_1, σ_2 and σ_3. If there is a collision for σ_1, σ_2 and σ_3, $\mathcal{A}$ can forge Bob as an Alice for the possession of the past session key, forge Alice as Bob for having the same session key and forge Bob for the key confirmation. According to the birthday paradox, the number of queries to the hash table $\mathcal{H}$ need to be at least $\frac{q_{hash}^2}{2^\ell}$. We use the difference lemma to show that

$$| Pr[S_1] - Pr[S_0] | \leq \frac{q_{hash}^2}{2^\ell}.$$

Game G_2: This game and the previous game are identical. This game differs by the fact that $\mathcal{A}$ tries to replace the ciper text, $\{GK^*, \lambda, ver_{gk}\}$. As the encryption algorithm holds IND-CPA, $\mathcal{A}$ needs at least $\frac{q_{Enc}^2}{2^k}$ queries to the $\mathcal{E}$ according to the birthday paradox. As per the difference lemma, the advantage of $\mathcal{A}$ in this game is

$$| Pr[S_2] - Pr[S_1] | \leq \frac{q_{Enc}^2}{2^k}.$$

Game G_3: This game is identical to the previous game and presents the KKS. $\mathcal{A}$ utilises the queries: $Session - state\ reveal(\mathcal{P}_{Alice,Bob}^{(\neg i)})$ and $Session - key\ reveal(\mathcal{P}_{Alice,Bob}^{(\neg i)})$ without violating the freshness of the ith session. Despite having remaining session keys and secrets of the other sessions, $\mathcal{A}$ requires a and λ of the ith session, which is chosen at random for every session and, the long-term secrets x_A, x_B, δ and K_c to know GK^*. But $\mathcal{A}$ does not need all the secrets at a stretch. The need for the different combinations of secrets leads to the following cases.

Case i: $\mathcal{A}$ needs either the product $a \cdot H(x_A \parallel \delta \parallel ID)$ or $H(x_B \parallel \delta \parallel ID)$ alone to get GK^*. That is, knowing $a \cdot H(x_A \parallel \delta \parallel ID)$ or $H(x_B \parallel \delta \parallel ID)$, SK can be computed. $a \cdot H(x_A \parallel \delta \parallel ID)$ or $H(x_B \parallel \delta \parallel ID)$ can be obtained if $\mathcal{A}$ employs an ECDLP solver by feeding $PKV_{Alice(i)}$ or $PKV_{Bob(i)}$, respectively. Alternatively, $H(x_A \parallel \delta \parallel ID)$ or $H(x_B \parallel \delta \parallel ID)$ can be obtained if $\mathcal{A}$ queries $\mathcal{H}$. a can be replaced with a random number. The advantage of $\mathcal{A}$ for employing the solver is Adv_{ECDLP}, the advantage for querying $\mathcal{H}$ is $\frac{q_{hash}^2}{2^l}$. The advantage gained by $\mathcal{A}$ in replacing the random number is $\frac{1}{2^m}$. Therefore, the total advantage of $\mathcal{A}$ is $\max(\frac{q_{hash}^2}{2^l}, \frac{q_{hash}^2}{2^l} + \frac{1}{2^m}, Adv_{ECDLP})$.

Case ii: $\mathcal{A}$ can convince Bob for the validity of its certificate (Step 6) if $\mathcal{A}$ solves ECq-SDHP. $\mathcal{A}$ can inject its own public key by forging $Cert_{Alice(i)}^*$. Similarly, $\mathcal{A}$ can convince Alice. To solve ECq-SDHP, $\mathcal{A}$ deploys an ECq-SDHP solver by feeding $Cert_{Alice(i-1)}^*$. The advantage of $\mathcal{A}$ in the solving is $Adv_{ECq-SDHP}$.

Case iii: $\mathcal{A}$ can compute SK if it solves ECDHP. For solving ECDHP, $\mathcal{A}$ employs a solver and gives $a \cdot PKV_{Alice(i)}^*$ and $PKV_{Bob(i)}^*$. The advantage of $\mathcal{A}$ for the employment is Adv_{ECDHP}.

Case iv: $\mathcal{A}$ can acquire GK^* if it breaks IND-CPA for $\{\lambda, ver_{gk}, GK^*\}SK$. For breaking the indistinguishability, $\mathcal{A}$ needs at least $\frac{q_{Enc}^2}{2^k}$ queries to $\mathcal{E}$ as per the birthday paradox. Therefore, the advantage of $\mathcal{A}$ is $\frac{q_{Enc}^2}{2^m}$, which is equal to the advantage of $\mathcal{A}$ in G_2.

Case v: $\mathcal{A}$ can construct GK^* if it knows ith session λ, K_c and δ. λ can be replaced with advantage $\frac{1}{2^m}$. K_c and δ can be replaced by a random number that is drawn from the uniform distribution with advantage Adv_{sk}. The total advantage of $\mathcal{A}$ is $(\frac{1}{2^m} + Adv_{sk})$.

The net advantage of $\mathcal{A}$ in this game is the maximum probability in any one of the above cases. We use the difference lemma to show that

$$| Pr[S_3] - Pr[S_2] | \leq \max\{\max\{\frac{q_{hash}^2}{2^l}, \frac{q_{hash}^2}{2^l} + \frac{1}{2^m}, Adv_{ECDLP}\},$$
$$Adv_{ECq-SDHP}, Adv_{ECDHP}, \frac{q_{Enc}^2}{2^m}, \frac{1}{2^m} + Adv_{sk}\}\}.$$

Game G_4: This game resembles G_3. $\mathcal{A}$ needs the secrets to identify either Alice or Bob and link their communication. $\mathcal{A}$ issues $Session - state\ reveal(\mathcal{P}_{Alice,Bob}^{(\neg i)})$ without losing the freshness of the ith session. That is, $\mathcal{A}$ issues the query to remaining sessions to acquire a and λ of all the sessions except the ith session. Although $\mathcal{A}$ holds the session-specific secrets of various sessions, it requires δ and x of any session and ith session to violate anonymity and unlinkability. The requirement gives two cases: $\mathcal{A}$ can target either Alice or Bob, or it can target both.

Case (i): $\mathcal{A}$ issues $Session - state\ reveal(\mathcal{P}^{(\neg i)})$ to either Alice or Bob and gets remaining session-specific secrets. In particular, if Alice is targeted, then the remaining session-specific secret a can be acquired. If Bob is targeted, then the secret λ can be acquired by $\mathcal{A}$. But $\mathcal{A}$ can trace neither Alice nor Bob by knowing λ and a. To trace either Alice or Bob, δ is mandatory for $\mathcal{A}$, which can be replaced with advantage Adv_{sk}, and x of the ith session and remaining session that can be replaced with advantage Adv_{sk}^2. The total advantage of $\mathcal{A}$ is $Adv_{sk}^2 + Adv_{sk}$.

Case (ii): $\mathcal{A}$ issues $Session - state\ reveal(\mathcal{P}^{(\neg i)})$ to both Alice and Bob. As $\mathcal{A}$ targets both Alice and Bob, it requires x of the ith session and remaining session for both Alice and Bob. δ can be acquired with advantage Adv_{sk}, and the long-term secrets of Alice and Bob can be replaced with advantage $2 \cdot Adv_{sk}^2$. The total advantage of $\mathcal{A}$ is $2 \cdot Adv_{sk}^2 + Adv_{sk}$. We use the difference lemma to present the total advantage of $\mathcal{A}$ in this game

$$| Pr[S_4] - Pr[S_3] | \leq 3 \cdot Adv_{sk}^2 + 2 \cdot Adv_{sk}.$$

Game G_5: This game presents the PFS of $\mathcal{P}$. $\mathcal{A}$ can ask either $Corrupt(\mathcal{P}_{Alice}^{(i)})$ or $Corrupt(\mathcal{P}_{Bob}^{(i)})$ for obtaining the long-term secrects, namely x_A of Alice and x_B of Bob. The condition for issuing the queries is that $\mathcal{A}$ should not violate the freshness of the session. That is, if $Corrupt(\mathcal{P}_{Alice}^{(i)})$ is executed by $\mathcal{A}$, then $Remover(\mathcal{P}_{Bob}^{(i)})$ is executed to expire Bob and vice-versa. This means that $\mathcal{A}$ can find a matching partner for the ith session.

Case (i): $\mathcal{A}$ issues $Corrupt(\mathcal{P}_{Alice}^{(i)})$ and obtains x_A^*. Though $\mathcal{A}$ acquires the long-term secret, it can not obtain the key of ith session because $\mathcal{A}$ needs $a \cdot H(x_A^* \parallel \delta \parallel ID)$ of ith session to compute SK. The adversary can get $a \cdot H(x_A^* \parallel \delta \parallel ID)$ if it solves $ECDLP$. To solve $ECDLP$, $\mathcal{A}$ employs an ECDLP solver, and it feeds $a.PKV_{Alice(i)}$. The solver outputs $a \cdot H(x_A^* \parallel \delta \parallel ID)$ with advantage Adv_{ECDLP}.

Case (ii): $\mathcal{A}$ issues $Corrupt(\mathcal{P}_{Bob}^{(i)})$ to obtain x_B^*. Though $\mathcal{A}$ acquires x_B, it can not obtain the key of ith session because $\mathcal{A}$ needs $H(x_B^* \parallel \delta \parallel ID)$. $H(x_B^* \parallel \delta \parallel ID)$ can be attained by $\mathcal{A}$ in two ways: solving $ECDLP$ and querying $\mathcal{H}$. The advantage in the solving and the querying are Adv_{ECDLP} and $\frac{q_{hash}^2}{2^l}$, correspondingly. The advantage of $\mathcal{A}$ is $\max(\frac{q_{hash}^2}{2^l}, Adv_{ECDLP})$. According to the difference lemma, the advantage gained by $\mathcal{A}$ in this game is

$$| Pr[S_5] - Pr[S_4] | \leq \max\{Adv_{ECDLP}, \max(\frac{q_{hash}^2}{2^l} + Adv_{ECDLP})\}$$

$$Pr[S_5] = \frac{1}{2}.$$

It is apparent that the winning probability of $\mathcal{A}$ in the last game is $\frac{1}{2}$. By the triangular inequality, we have:

$$\begin{aligned}
&| Pr[S_0] - Pr[S_5] | = | Pr[S_0] - Pr[S_1] + Pr[S_1] - Pr[S_2] \\
&+ Pr[S_2] - Pr[S_3] + Pr[S_3] - Pr[S_4] \\
&+ Pr[S_4] - Pr[S_5] | \\
&\leq | Pr[S_0] - Pr[S_1] | + | Pr[S_1] - Pr[S_2] | + \\
&| Pr[S_2] - Pr[S_3] | + | Pr[S_3] - Pr[S_4] | + \\
&| Pr[S_4] - Pr[S_5] | .
\end{aligned}$$

This means that

$$\begin{aligned}
Adv_{\mathcal{P}}^{SS} \leq \; & \frac{q_{hash}^2}{2^\ell} + \frac{q_{Enc}^2}{2^k} + \\
& \max\{\max\{\frac{q_{hash}^2}{2^\ell}, \frac{q_{hash}^2}{2^\ell} + \frac{1}{2^m}, Adv_{ECDLP}\}, \\
& Adv_{ECq-SDHP}, Adv_{ECDHP}, \\
& \frac{q_{Enc}^2}{2^m}, \frac{1}{2^m} + Adv_{sk}\}\} + 3 \cdot Adv_{sk}^2 + 2 \cdot Adv_{sk} + \\
& \max\{Adv_{ECDLP}, \max(\frac{q_{hash}^2}{2^\ell} + Adv_{ECDLP})\}.
\end{aligned}$$

This concludes our proof. □

References

[1] D. Geronimo, A.M. Lopez, A.D. Sappa, T. Graf, Survey of pedestrian detection for advanced driver assistance systems, IEEE Trans. Pattern Anal. Mach. Intell. 32 (7) (2010) 1239–1258, https://doi.org/10.1109/TPAMI.2009.122.

[2] A. Bisoffi, F. Biral, M.D. Lio, L. Zaccarian, Longitudinal jerk estimation of driver intentions for advanced driver assistance systems, IEEE/ASME Trans. Mechatron. 22 (4) (2017) 1531–1541, https://doi.org/10.1109/TMECH.2017.2716838.

[3] V. Milanés, S.E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, M. Nakamura, Cooperative adaptive cruise control in real traffic situations, IEEE Trans. Intell. Transp. Syst. 15 (1) (2014) 296–305, https://doi.org/10.1109/TITS.2013.2278494.

[4] M.N. Mejri, J. Ben-Othman, M. Hamdi, Survey on vanet security challenges and possible cryptographic solutions, Veh. Commun. 1 (2) (2014) 53–66.

[5] J. Larson, K.Y. Liang, K.H. Johansson, A distributed framework for coordinated heavy-duty vehicle platooning, IEEE Trans. Intell. Transp. Syst. 16 (1) (2015) 419–429, https://doi.org/10.1109/TITS.2014.2320133.

[6] M. Azees, P. Vijayakumar, L.J. Deboarh, Eaap: efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks, IEEE Trans. Intell. Transp. Syst. 18 (9) (2017) 2467–2476, https://doi.org/10.1109/TITS.2016.2634623.

[7] D. Jia, K. Lu, J. Wang, X. Zhang, X. Shen, A survey on platoon-based vehicular cyber-physical systems, IEEE Commun. Surv. Tutor. 18 (1) (2016) 263–284, https://doi.org/10.1109/COMST.2015.2410831.

[8] H. Hartenstein, L.P. Laberteaux, A tutorial survey on vehicular ad hoc networks, IEEE Commun. Mag. 46 (6) (2008) 164–171, https://doi.org/10.1109/MCOM.2008.4539481.

[9] M. Raya, P. Papadimitratos, J.p. Hubaux, Securing vehicular communications, IEEE Wirel. Commun. 13 (5) (2006) 8–15, https://doi.org/10.1109/WC-M.2006.250352.

[10] N.W. Lo, H.C. Tsai, Illusion attack on vanet applications - a message plausibility problem, in: 2007 IEEE Globecom Workshops, 2007, pp. 1–8.

[11] F. Wang, Y. Xu, H. Zhang, Y. Zhang, L. Zhu, 2flip: a two-factor lightweight privacy-preserving authentication scheme for vanet, IEEE Trans. Veh. Technol. 65 (2) (2016) 896–911.

[12] J.L. Huang, L.Y. Yeh, H.Y. Chien, Abaka: an anonymous batch authenticated and key agreement scheme for value-added services in vehicular ad hoc networks, IEEE Trans. Veh. Technol. 60 (1) (2011) 248–262, https://doi.org/10.1109/TVT.2010.2089544.

[13] T.W. Chim, S.-M. Yiu, L.C. Hui, V.O. Li, Specs: secure and privacy enhancing communications schemes for vanets, Ad Hoc Netw. 9 (2) (2011) 189–203.

[14] S.-J. Horng, S.-F. Tzeng, Y. Pan, P. Fan, X. Wang, T. Li, M.K. Khan, b-specs+: batch verification for secure pseudonymous authentication in vanet, IEEE Trans. Inf. Forensics Secur. 8 (11) (2013) 1860–1875.

[15] B. Ying, A. Nayak, Anonymous and lightweight authentication for secure vehicular networks, IEEE Trans. Veh. Technol. 66 (12) (2017) 10626–10636, https://doi.org/10.1109/TVT.2017.2744182.

[16] K. Abboud, H.A. Omar, W. Zhuang, Interworking of dsrc and cellular network technologies for v2x communications: a survey, IEEE Trans. Veh. Technol. 65 (12) (2016) 9457–9470.

[17] Y.H. Park, S.W. Seo, Fast and secure group key dissemination scheme for out-of-range v2i communication, IEEE Trans. Veh. Technol. 64 (12) (2015) 5642–5652, https://doi.org/10.1109/TVT.2015.2487832.

[18] A. Ornaghi, M. Valleri, Man in the middle attacks Demos, Blackhat [Online Document] 19 (2003).

[19] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: Proceedings of the 1st ACM Conference on Computer and Communications Security, ACM, 1993, pp. 62–73.

[20] D. Huang, S. Misra, M. Verma, G. Xue, Pacp: an efficient pseudonymous authentication-based conditional privacy protocol for vanets, IEEE Trans. Intell. Transp. Syst. 12 (3) (2011) 736–746.

[21] 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, https://doi.org/10.1109/TITS.2015.2439292.

[22] J.-S. Li, K.-H. Liu, A lightweight identity authentication protocol for vehicular networks, Telecommun. Syst. 53 (4) (2013) 425–438.

[23] Z. Zhou, H. Zhang, Z. Sun, An improved privacy-aware handoff authentication protocol for vanets, Wirel. Pers. Commun. 97 (3) (2017) 3601–3618.

[24] L. Zhang, Q. Wu, A. Solanas, J. Domingo-Ferrer, A scalable robust authentication protocol for secure vehicular communications, IEEE Trans. Veh. Technol. 59 (4) (2010) 1606–1617.

[25] A. Menezes, An introduction to pairing-based cryptography, Recent Trends Cryptogr. 477 (2009) 47–65.

[26] V.S. Miller, Use of elliptic curves in cryptography, in: Conference on the Theory and Application of Cryptographic Techniques, Springer, 1985, pp. 417–426.

[27] F. Vercauteren, Optimal pairings, IEEE Trans. Inf. Theory 56 (1) (2009) 455–461.

[28] D. Boneh, X. Boyen, Short signatures without random oracles, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2004, pp. 56–73.

[29] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, Z. Zhang, Creditcoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles, IEEE Trans. Intell. Transp. Syst. 19 (7) (2018) 2204–2220.

[30] D.A. Rivas, J.M. Barceló-Ordinas, M.G. Zapata, J.D. Morillo-Pozo, Security on vanets: privacy, misbehaving nodes, false information and secure data aggregation, J. Netw. Comput. Appl. 34 (6) (2011) 1942–1955.

[31] M. Wolf, T. Gendrullis, Design, implementation, and evaluation of a vehicular hardware security module, in: International Conference on Information Security and Cryptology, Springer, 2011, pp. 302–318.

[32] J.B. Kenney, Dedicated short-range communications (dsrc) standards in the United States, Proc. IEEE 99 (7) (2011) 1162–1182.

[33] C. Boyd, A. Mathuria, Protocols for Authentication and Key Establishment, Springer Science & Business Media, 2013.

[34] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Trans. Parallel Distrib. Syst. 11 (8) (2000) 769–780, https://doi.org/10.1109/71.877936.

[35] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, Adv. Cryptol.—EUROCRYPT 2001 (2001) 453–474.

[36] J. Petit, F. Schaub, M. Feiri, F. Kargl, Pseudonym schemes in vehicular networks: a survey, IEEE Commun. Surv. Tutor. 17 (1) (2015) 228–255.

[37] S.S. Manvi, S. Tangade, A survey on authentication schemes in vanets for secured communication, Veh. Commun. 9 (2017) 19–30.

[38] M. Abdalla, P.-A. Fouque, D. Pointcheval, Password-based authenticated key exchange in the three-party setting, in: International Workshop on Public Key Cryptography, Springer, 2005, pp. 65–84.

[39] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated key exchange secure against dictionary attacks, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2000, pp. 139–155.
[40] V. Shoup, Sequences of games: a tool for taming complexity in security proofs, IACR Cryptol. ePrint Archive 2004 (2004) 332.
[41] Tamarin scripts of this paper: https://github.com/FUllSCIPTS/paper2, 2019.
[42] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (2) (1983) 198–208.
[43] S. Meier, B. Schmidt, C. Cremers, D. Basin, The tamarin prover for the symbolic analysis of security protocols, in: International Conference on Computer Aided Verification, Springer, 2013, pp. 696–701.
[44] Tamarin-Prover Manual Security Protocol Analysis in the Symbolic Model, https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf, 2019.
[45] G. Lowe, A hierarchy of authentication specifications, in: Computer Security Foundations Workshop, 1997. Proceedings, 10th, IEEE, 1997, pp. 31–43.
[46] U. of Tsukuba, Tepla: elliptic curve and pairing library, Tech. Rep., University of Tsukuba, 2013, http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html.
[47] https://pypi.org/project/pycrypto/, 2020.
[48] https://pypi.org/project/tate_bilinear_pairing/, 2020.