Journal of Information Security and Applications 74 (2023) 103476

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

Blockchain-based secret key extraction for efficient and secure authentication in VANETs

Mahmoud A. Shawky [a,*], Muhammad Usman [b], David Flynn [a], Muhammad Ali Imran [a], Qammer H. Abbasi [a], Shuja Ansari [a,*], Ahmad Taha [a,*]

[a] James Watt School of Engineering, University of Glasgow, G12 8QQ, Glasgow, UK
[b] School of Computing, Engineering and Built Environment, Glasgow Caledonian University, G4 0BA, Glasgow, UK

ARTICLE INFO

Keywords:
AVISPA simulation
BAN-logic
Key reconciliation
Public key infrastructure
Secret key extraction
Smart contracts-based blockchain

ABSTRACT

Intelligent transportation systems are an emerging technology that facilitates real-time vehicle-to-everything communication. Hence, securing and authenticating data packets for intra- and inter-vehicle communication are fundamental security services in vehicular ad-hoc networks (VANETs). However, public-key cryptography (PKC) is commonly used in signature-based authentication, which consumes significant computation resources and communication bandwidth for signatures generation and verification, and key distribution. Therefore, physical layer-based secret key extraction has emerged as an effective candidate for key agreement, exploiting the randomness and reciprocity features of wireless channels. However, the imperfect channel reciprocity generates discrepancies in the extracted key, and existing reconciliation algorithms suffer from significant communication costs and security issues. In this paper, PKC-based authentication is used for initial legitimacy detection and exchanging authenticated probing packets. Accordingly, we propose a blockchain-based reconciliation technique that allows the trusted third party (TTP) to publish the correction sequence of the mismatched bits through a transaction using a smart contract. The smart contract functions enable the TTP to map the transaction address to vehicle-related information and allow vehicles to obtain the transaction contents securely. The obtained shared key is then used for symmetric key cryptography (SKC)-based authentication for subsequent transmissions, saving significant computation and communication costs. The correctness and security robustness of the scheme are proved using Burrows–Abadi–Needham (BAN)-logic and Automated Validation of Internet Security Protocols and Applications (AVISPA) simulator. We also discussed the scheme's resistance to typical attacks. The scheme's performance in terms of packet delay and loss ratio is evaluated using the network simulator (OMNeT++). Finally, the computation analysis shows that the scheme saves ~ 99% of the time required to verify 1000 messages compared to existing PKC-based schemes.

1. Introduction

Intelligent transportation systems aim to provide drivers with real-time traffic information to avoid road congestion and potential collisions, thus maximizing traffic efficiency [1]. Due to the rapid development of wireless communication technology, vehicular ad-hoc network (VANET) has emerged in many traffic applications such as safety, autonomy, navigation, etc [2]. VANETs typically offer two types of vehicular communications, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) [1,2]. For safety-related applications, vehicles wirelessly communicate with nearby terminals (e.g., vehicles and infrastructures) based on the IEEE 802.11P standard [3]. In this context, a safety-related message is sent within a period of 100–300 ms using the dedicated short-range communication (DSRC) protocol in the frequency band from 5.85 to 5.925 GHz [4]. However, the public accessibility of wireless channels makes these messages susceptible to interception, modification, and fabrication [5]. Thus, vehicular communication must be secured through authentication, as a critical security service. VANETs architecture typically includes a trusted authority (TA), roadside units (RSUs), and vehicles' onboard units (OBUs) [6].

The current state-of-the-art for authentication in VANETs is categorized based on: cryptography, signature, and verification methodologies, as shown in Fig. 1 [1]. For cryptography-based authentication, public key infrastructure-based (PKI-based) and identity-based (ID-based) approaches are the most conventional authentication

* Corresponding authors.
E-mail addresses: m.shawky.1@research.gla.ac.uk (M.A. Shawky), muhammad.usman@gcu.ac.uk (M. Usman), David.Flynn@gla.ac.uk (D. Flynn), Muhammad.Imran@gla.ac.uk (M.A. Imran), Qammer.Abbasi@gla.ac.uk (Q.H. Abbasi), Shuja.Ansari@gla.ac.uk (S. Ansari), Ahmad.Taha@gla.ac.uk (A. Taha).

https://doi.org/10.1016/j.jisa.2023.103476

Available online 28 March 2023

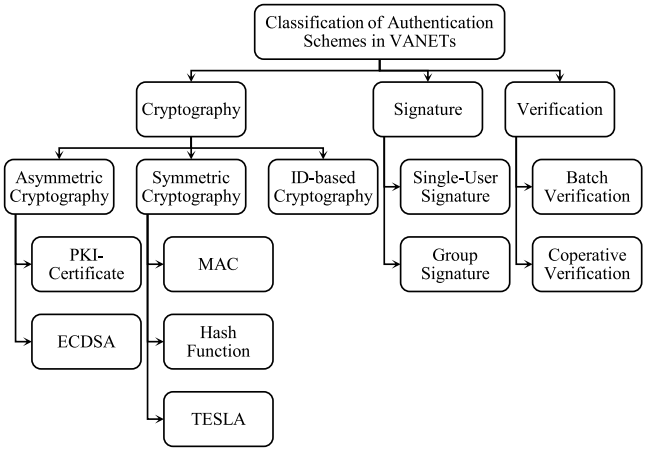


Fig. 1. Classification of authentication in VANETs [1].

methodologies [2]. However, group signature-based (GS-based) authentication is a commonly adopted signature-based authentication approach to support privacy preservation. In PKI-based approaches, a digital certificate is signed and issued by the TA to prove the sender's legitimacy and certify that the attached public key belongs to a certain user in the network [7]. In ID-based approaches, the user's identity information is used to derive the public key, while the private key is computed and distributed by the key generation center (i.e., TA) based on the given identity information [8]. By doing so, the receiver verifies messages using the sender's public key while signing it using its private key. In GS-based approaches, vehicles sign messages on behalf of the group without revealing their identities [9]; only the TA, as a group manager, can trace vehicles' real identities. Nevertheless, most of the existing VANETs' authentication schemes commonly use public key cryptography (PKC) to generate and verify signatures, which consumes significant computation costs compared to symmetric key cryptography (SKC) [10]. For SKC-based authentication, the key management process constitutes a challenging issue, particularly given the short-term communication sessions of high mobility terminals.

Recently, physical (PHY)-layer security has emerged as a computationally efficient and innovative method of securing key agreement between communicating terminals, exploiting the short-term channel reciprocity and randomness to extract a secret cryptographic shared key [11]. The key extraction process relies on the wireless channels' spatial and temporal variations between legitimate terminals. Hence, an eavesdropper cannot establish a rational relationship between different extracted keys from different sessions, maintaining forward and backward secrecy. In addition, this approach helps in detecting and mitigating Sybil attacks (in which attackers masquerade as multiple innocent vehicles [12]) since an adversary with multiple fabricated identities has highly correlated channel observations within the same coherence period.

The channel's unpredictable responses (i.e., received signal strength and phase) are used as a natural source of randomness to extract high entropy secret keys [13]. By probing the channel and having the channel estimates within the coherence period T_c, the obtained estimates undergo three main stages, i.e., quantization, information reconciliation, and privacy amplification — see Fig. 2 [14]. The quantization stage is a mapping operation that converts the channel components into bit streams. While the information reconciliation stage is an error correction stage that involves correcting the mismatched bits resulting from the imperfect channel reciprocity. The final stage utilizes a hashing operation to maintain the secrecy of the extracted key. One of the challenges of the secret key extraction process is the significant communication cost incurred by the reconciliation stage [10]. Furthermore, a reconciliation approach such as the Cascade algorithm exposes

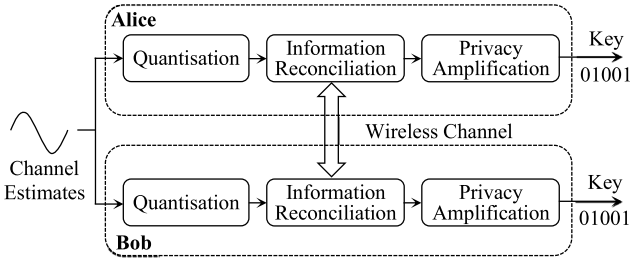


Fig. 2. PHY-layer-based secret key extraction mechanism.

60% of the matched bits to reconcile only 10% of the mismatched bits, posing a security threat [10]. Other reconciliation approaches, such as low-density parity-check [15] and turbo [16] codes, suffer from high computation complexities [11]. Accordingly, this paper addresses these limitations by designing a blockchain-based reconciliation technique that allows a trusted third party (TTP) to serve as a referee between the communicating vehicles by publishing a transaction containing the correction sequence (CS) of the mismatched bits using smart contract-based blockchain technology. The published transaction allows the vehicles to obtain the CS while the transaction address serves as temporary proof of trustworthiness for the entire session rather than transmitting a certificate every time, thereby saving communication costs and storage capacity.

In terms of key extraction, several theoretical approaches have been published [14]. However, the complexities involved in their practical integration with SKC-based applications are typically overlooked. This study presents a blockchain-based authentication scheme in which a PKI-based approach is used for handshaking between communicating vehicles and the exchanging of authenticated probing packets. After the quantization of the channel estimates, the CS is published by the TTP in the blockchain to address the discrepancies and have a secret shared key to be used for subsequent transmissions. For all subsequent transmissions, we use SKC, the advanced encryption standard AES algorithm for re-authentication, saving significant computation costs over PKC-based methods.

The following summarizes this paper's contributions.

1. We propose a blockchain-based secret key extraction (BCSKE) scheme for authentication in VANETs. The BCSKE scheme incorporates the channel phase response-based secret key extraction algorithm [13] for key agreement between communicating terminals. Accordingly, the proposed scheme uses PKC and SKC-based signatures at first and subsequent transmissions, respectively, mitigating the significant costs of using PKC-based signatures for each transmission.
2. We demonstrated how the smart contract can be used to establish and publish the relationship between CS and vehicles' related information via a transaction, hence, leveraging the immutable and memorable properties of blockchain technology to map the transaction address to vehicles' related information.
3. We demonstrated the correctness and security robustness of the BCSKE scheme using Burrows–Abadi– Needham (BAN)-logic analysis and Automated Validation of Internet Security Protocols and Applications (AVISPA) simulation tool. Our discussion also covered the scheme's resistance to various attacks.
4. Finally, we analyzed the scheme's performance and conducted a comprehensive evaluation in terms of computation and communication costs, authentication delay, and packet loss ratio using the OMNeT++ network simulator.

The remainder of this paper is organized as follows. Section 2 reviews the state-of-the-art in authentication in VANETs. Section 3 demonstrates the preliminary aspects of the BCSKE scheme. details

the steps involved in developing the proposed scheme. Sections 5 and 6 evaluate the scheme's security strength and performance, respectively. Finally, Section 7 presents the primary findings and conclusions generated from this work.

2. Related works

This section provides an overview of existing work relating to authentication and blockchain in VANETs. For simplicity, Table 1 lists the notations used in this section.

2.1. An overview of authentication in VANETs

This subsection provides a comprehensive overview of the commonly used authentication methods in VANETs and highlights the motivation for considering the use of SKC for lightweight authentication. Many studies have been presented to the research community to support the security and privacy requirements of VANETs. In [7], Raya et al. proposed a modified PKI-based approach, wherein a large number of digital certificates and their associated key pairs are preloaded onto the vehicle's OBUs. During each transmission, the vehicle randomly selects a key pair for message signing and verification, supporting unlinkability. In [17–20], the authors proposed conditional privacy-preserving authentication (CPPA) schemes in which signatures are generated and verified using elliptic curve cryptosystem (ECC)-based scalar multiplications and additions operations. According to [18], a pseudo-ID-based scheme is proposed in which pseudo-identities are exchanged between terminals to offer conditional privacy. In [21,22], the authors proposed certificate-less authentication methods, supporting VANETs' security and privacy objectives.

Other studies have been conducted to improve network scalability. Liu et al. [23] and Asaar et al. [24] proposed proxy vehicle-based authentication schemes. They designed an ID-based solution that employs vehicles' computational availability to verify signatures in favor of RSUs, mitigating the overhead on RSUs. For 6G-enabled VANET, Vijayakumar et al. [25] proposed an anonymous authentication and key exchange protocol. Chen et al. [26] use the time-consuming bilinear pairing for mutual authentication. Shao et al. [27] and Azees et al. [28] presented a GS-based solution using short-term anonymous certificates and bilinear pairing (BP) operations. In [27], the scheme was designed to support batch verification. The GS-based solution presented by Lim et al. [29] reduces the overhead on TA by dividing RSUs into leaders and members RSUs and giving leader RSUs the authority to generate group keys. Jiang et al. [30] employed region trust authorities (RTAs) to provide vehicles with efficient authentication services and decrease the overhead on TA and RSUs. Vijayakumar et al. [31] developed a dual authentication scheme based on group key distribution in VANETs. In this scheme, the key management process can effectively distribute and update the group's key for vehicles joining or leaving the group communication region. Xiong et al. [32] employed the chinese remainder theorem for group key dissemination across all vehicles in the same domain.

The previously mentioned studies have utilized PKC for authentication. However, this approach is characterized by a high computational load for signature verification. This is a limiting factor in terms of scalability, especially in scenarios involving a high number of vehicles. For example, in a scenario where 600 vehicles are within the communication range of a roadside unit or another vehicle, the endpoint terminal must verify 2000–6000 signatures/s, considering the transmission rate via the DSRC protocol to be 100 to 300 ms. This high number of signatures poses a challenge for PKC-based authentication and highlights the need for a more lightweight solution to reduce the computational demands for verifying massive crypto-based signatures in real-time. In this context, SKC represents an effective method, as it enables improvement in network scalability by reducing the computation required for signature verification.

Table 1
Scheme notations.

Symbol	Definition
Sk_{V_i}	The private key of the vehicle V_i
Pk_{V_i}	The public key of the vehicle V_i
T_R	The expiry date of the digital certificate
$Cert_{V_i}$	The digital certificate of the vehicle V_i
σ_i	The generated signature of the content x
T_i	The timestamp of the generated signature
T_r	The signature receiving time
T_Δ	The timestamp expiry period [00:00:59]
$T_{Session}$	The session expiry period [00:04:59]
$\hat{k}_{V_i}$	The extracted secret key by vehicle V_i
Sk_{RSU_j}	The private key of the RSU_j
Pk_{RSU_j}	The public key of the RSU_j
Sk_{RTA}	The private key of the RTA
Pk_{RTA}	The public key of the RTA
Sk_{TA}	The private key of the TA
Pk_{TA}	The public key of the TA
$CS_{V_{1-2}}$	The correction sequence of $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$
$TxID$	The transaction ID in the blockchain
T_{Tx}	The transaction publishing timestamp
$\|\|$	Concatenation between two variables

2.2. Blockchain-based authentication in VANETs

This subsection presents an overview of prevalent block-chain-based authentication techniques in VANETs and highlights the significance of proposing a blockchain-assisted key reconciliation approach. Lu et al. [33] employed the blockchain to design a proof of presence and absence of certificate issuance and revocation, respectively, offering conditional identity anonymity. In their scheme, a reputation score is sent with each transmission, indicating the degree of trustworthiness of the sender. Despite that, this solution cannot support unlinkability since the reputation scores are updated gradually. Thus, adversaries can trace broadcasted messages to build location-tracking attacks. Zheng et al. [34] adopted the blockchain for pseudo-ID-based authentication. In this scheme, the blockchain acts as a storage and transparency mechanism for secure and decentralized transactions. Ogundoyin et al. [35] introduce a decentralized and transparent revocation process using two blockchains. Lu et al. [36] combined the Merkle Patricia Tree with the blockchain to enable monitoring of the authority's activities, thus promoting transparency. However, the process of generating anonymous certificates requires frequent interactions between vehicles and the certificate authority. Ma et al. [37] employed the bivariate polynomial function to design a decentralized key management process. In this scheme, the vehicle service provider updates vehicles' expiring private and public keys using smart contracts, securing the vehicles from DoS and collusion attacks. Azees et al. [38] introduced a blockchain-based anonymous authentication scheme for VANET communication.

Readers interested in this topic are referred to lin et al. [39], where the authors integrated the blockchain "Ethereum" technology into the PKI-based approach to present a certificate distribution and revocation mechanism. In transactions, the CA updates the blockchain with users' public key certificates' blocks, allowing network terminals to securely verify the received signatures via transactions' addresses as proof of activities. Son et al. [40] suggested a consortium blockchain-based handover authentication protocol for V2I communication. This scheme offers an efficient initial authentication using the scalar multiplication of the elliptic curve cryptosystem in conjunction with the computationally inexpensive hashing operation. Nonetheless, the proposed scheme cannot defend against de-synchronization attacks. In this study, a novel blockchain application is presented which aims to reconcile the mismatched bits from the PHY-layer secret key extraction process through the use of smart contracts. Besides, the published transactions work as proof of trustworthiness for all subsequent transmissions.

In this context, blockchain protocols must be resistant to birthday collisions and hijackings. By minimizing the possibility of generating

Table 2
A comparison of different blockchain types [41].

Feature	Public	Private	Consortium
Ownership	Nobody	Centralized entity	Multiple entities
Joining eligibility	Free to join	Permissioned/approved participants	Permissioned/approved participants
Consensus type	Proof of Stake (PoS)/Proof of Work (PoW)	Voting or multi-party consensus algorithm	Voting or multi-party consensus algorithm
Transaction speed	Not fast as Private/Consortium	Light and fast	Light and fast
Decentralization	Decentralized	Less decentralized	Less decentralized

two of the same blocks simultaneously, the protocol can effectively prevent birthday collisions. In addition, an attacker must not be able to hijack transactions by hijacking the protocol (i.e., the protocol must ensure the non-modifiability of transactions) [39]. Generally, there are three categories of blockchain networks based on their uses and applications' requirements: public, private, and consortium. Table 2 shows a comparison of different blockchain types [41]. It can be noted from Table 2 that any terminal can join and interact with the public blockchain. By doing so, the network entities can verify and monitor transactions published by centralized authorities, thus promoting transparency. However, public blockchain is not as fast as private and consortium blockchain networks for approving transactions. It is considered a trade-off relationship between transparency and latency. Consequently, the proposed scheme is implemented within the Ethereum-based public blockchain network with the aim of promoting transparency.

3. Key extraction and system model

In this section, we review the key extraction algorithm in [13]. Then, the system modeling is discussed in detail.

3.1. Review of the channel phase response-based secret key extraction algorithm in [13]

The pairwise key extraction process between communicating vehicles, V_1 and V_2, consists of the following steps.

- *Step 1*: V_1 sends V_2 a probing packet PP_{V_1} at time T_1 in the following simplified form.

$$PP_{V_1}(T_1) = e^{j(w_c T_1 + \phi_1)} \tag{1}$$

where ϕ_1 is a uniformly distributed random phase chosen by V_1 within the interval $[0, 2\pi)$. So that V_2's received signal can be formulated as

$$R_{V_{12}}(t) = \alpha_{12} e^{j(w_c t + \phi_1 + \theta_{12})} + \eta_{12}(t) \tag{2}$$

where $\eta_{12}(t)$ is the additive white gaussian noise (AWGN) and α_{12} and θ_{12} are the forward link channel gain and phase responses, respectively. At last, V_2 obtains the noisy phase estimate $\hat{\phi}_{12} \approx \phi_1 + \theta_{12}$.
- *Step 2*: Similarly, V_2 sends V_1 a probing packet PP_{V_2} at time T_2 in the following simplified form.

$$PP_{V_2}(T_2) = e^{j(w_c T_2 + \phi_2)} \tag{3}$$

where ϕ_2 is a uniformly distributed phase chosen by V_2 within the interval $[0, 2\pi)$. So that V_1's received signal can be formulated as

$$R_{V_{21}}(t) = \alpha_{21} e^{j(w_c t + \phi_2 + \theta_{21})} + \eta_{21}(t) \tag{4}$$

where $\eta_{21}(t)$ is the AWGN and α_{21} and θ_{21} are the reverse link channel gain and phase responses, respectively. At last, V_1 obtains the noisy phase estimate $\hat{\phi}_{21} \approx \phi_2 + \theta_{21}$.
- *Step 3*: Both vehicles compute the final phase components used for the key extraction as follows.

$$\begin{aligned} V_1 &: \Phi_1 = \hat{\phi}_{21} + \phi_1 \bmod 2\pi \\ V_2 &: \Phi_2 = \hat{\phi}_{12} + \phi_2 \bmod 2\pi \end{aligned} \tag{5}$$

Note that, $\theta_{12} \approx \theta_{21}$ for $T_2 - T_1 \leq T_c$.
- *Step 4*: Finally, both vehicles map Φ_1 and Φ_2 into the quantization region to get $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$ by applying the following formula.

$$Q(x) = k \quad \text{if } x \in \left[\frac{2\pi(k-1)}{q}, \frac{2\pi k}{q}\right) \tag{6}$$

for $k = 1, 2, \ldots, q$. See Ref. [13] for more details.

3.2. System modeling

In the proposed BCSKE scheme, six entities are involved: the TA, the RTAs, the RSUs, the vehicles' OBUs, and the smart contract using blockchain technology — see Fig. 3. The following defines the role of the network entities.

- *TA*: The TA initializes the scheme public parameters and registers the network terminals in the system. It has the authority to reveal the real identities of the network terminals in case of malicious behaviors. Furthermore, it distributes the certificate revocation list (CRL) of the misbehaving vehicles between terminals.
- *RTA*: In each region, there is a RTA that provides vehicles with efficient authentication services and reduces the TA's computational overhead. The RTA's regional centralized servers are responsible for reconciling the mismatched bits of the extracted keys between vehicles by triggering the smart contract and publishing the computed correction sequences in the blockchain through transactions in an orderly fashion.
- *RSU*: In both directions of the road, the RSUs are deployed with high storage and computation capacities. It functions as a cooperative relay between vehicles and the RTA, allowing wireless and wired communication between itself and surrounding vehicles as well as between itself and the RTA, respectively.
- *OBUs*: It is a vehicle-mounted processing unit with constrained computation capabilities and a tamper-proof property. It also has the availability to trigger the smart contract's **Get** function and retrieve the transaction information from the blockchain.
- *Blockchain Network*: The Blockchain Network is a decentralized distributed database system that offers immutable, undeniable, and verifiable data storage through transactions. Recently, researchers have been contributing to the development of blockchain technology with horizontal and vertical scaling expansion, such as Ethereum and Hyperledger, respectively [42]. The horizontally expanded blockchain is a public blockchain since anyone can join or leave it. In contrast, the vertically expanded blockchain can only be joined by trusted nodes, so it is referred to as a private blockchain. Both types maintain an immutable and chronological sequence of chaining using proof-of-work (PoW) and proof-of-stack (PoS) consensus mechanisms. Our approach entails embedding the correction sequence of the shared key into the transaction such that vehicles can reconcile the mismatched bits of the shared key from the transaction content. RTAs' activities in VANET are transparent and verifiable, so the transaction contents in this study function as a short-term digital certificate for a specific period called the session time $T_{Session}$.
- *Smart contract (SC) using blockchain technology*: *SC*s are self-executing transactions-based contracts whose terms and conditions are written in the form of codes using the Turing complete scripting language (i.e., *Solidity* for Ethereum smart contracts). These codes are distributed throughout the decentralized

blockchain network. Based on the conditions assigned, the SC's built-in functions can be triggered. This paper uses the public Ethereum blockchain as the platform for the creation of the SC. Two main reasons behind the use of the SC in Algorithm (1) are to (I) publish the correction sequence of the extracted key's mismatched bits between the communicating vehicles, V_1 and V_2, via a transaction Tx, retrieving its address $TxID$, and (II) use the retrieved $TxID$ as a proof of trustworthiness for subsequent transmissions. There are four functions to be provided in the involved SC. The **Deployer** function is used to specify the address of the RTA (owner) that is authorized to deploy the SC in the blockchain. The **IssueCS**(Pk_{V_1}, Pk_{V_2}, CS) function can only be invoked by the RTA (only owner), which is used to publish the correction sequence along with the communicating vehicles' public keys (Pk_{V_1}, Pk_{V_2}) in the blockchain and get $TxID$. The **Update**$(Pk_{V_1}, Pk_{V_2}, TxID)$ function is used to map $TxID$ to (Pk_{V_1}, Pk_{V_2}), and also can only be invoked by the RTA. The **Get**(Pk_{V_1}, Pk_{V_2}) function is a view function that can be invoked by any network terminal to retrieve $TxID$ without incurring any gas fees.

Algorithm 1: Smart Contract for **KeyAgreement**

Given: function name, parameter settings

Require: Setting up functions

```
struct V2V {uint pk1; uint pk2; uint CS;
} //Define the input parameter types
address RTA = 0xcbB3012b86b594223E43FB9c56624F357463b;
mapping (uint → uint256) public PK2TX;
function Deployer ( ) public {RTA = msg.sender;
} //Define the deployer as the RTA
modifier onlyowner {require (msg.sender == RTA);
_;
}//Identify the message sender
V2V keyagreementl;
function IssueCS (uint _pk1, uint _pk2, uint _CS)
onlyowner public returns (uint, uint, uint) {
keyagreement1.pk1 = _pk1;
keyagreement1.pk2 = _pk2;
keyagreementl.CS = _CS;
return (keyagreement1.pk1, keygreement1.pk2, keyagreement1.CS)
} //Generate a transaction for CS and get TxID
function Update (uint pk1, uint pk2, uint256 TxID)
onlyowner public {
PK2TX [pk1 ∧ pk2] = TxID;
} //Mapping the inserted pair of public keys to TxID
function Get (uint pk1, uint pk2) public view returns
(uint256) {
return PK2TX [pk1 ∧ pk2];
} //Retrieve the TxID by vehicles
```

4. The proposed scheme

This section describes our BCSKE scheme implemented on the public blockchain (e.g., Ethereum) - see Fig. 4. In this scheme, each network terminal (i.e., vehicles and RSUs) possesses a long-term digital public key certificate that is used for initial legitimacy detection using PKI-based authentication. Taking advantage of the short-term reciprocal properties of the channel phase responses and employing its unpredictable behavior as a source of randomness, both vehicles, $V_1(Alice)$ and $V_2(Bob)$, can exchange authenticated and time-stamped probing packets PP within the coherence interval T_c to extract high entropy secret keys (4.3: Step 1~2). Due to the use of the half-duplex mode when probing the channel, the extracted bit sequences, $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$, have some discrepancies. The bit mismatch rate (BMR) is used to define

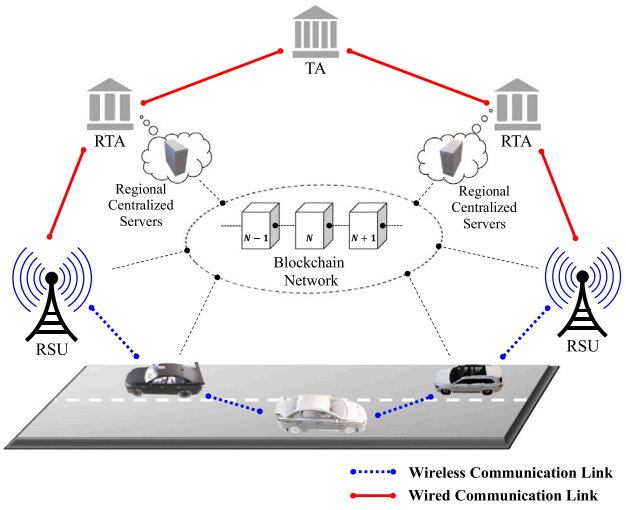


Fig. 3. VANET architecture using blockchain technology.

the number of mismatched bits to the total number of channel samples, formulated as

$$BMR = \frac{\bar{I}(\hat{k}_{V_1}, \hat{k}_{V_2})}{No.\ Channel\ Samples} \tag{7}$$

where $\bar{I}(\hat{k}_{V_1}, \hat{k}_{V_2})$ is the number of mismatched/incorrect bits between $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$. While the bit generation rate (BGR) is defined as the order/length of the extracted bit sequence to the total number of channel samples, denoted by

$$BGR = \frac{Bit\ Length\ (\hat{k}_{V_{1(2)}})}{No.\ Channel\ Samples} \tag{8}$$

In order to avoid the communication cost and security flaws associated with the information reconciliation stage, both vehicles encrypt $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$ and send them to the RTA within the same region (4.3: Step 3~6). In the proposed scheme, the RTA acts as a referee (TTP) between the communicating vehicles, correcting the mismatched bits and generating the correction sequence. After the RTA deploys the SC, it establishes the relationship between the pair of public keys of the communicating vehicles and its associated correction sequence (4.3: Step 7). Finally, both vehicles can obtain an identical shared key 4.4 used for symmetric key cryptography at subsequent transmissions (4.5: Step 1, 2). In general, the BCSKE scheme consists of five phases, i.e., system initialization, registration, initial verification and channel probing, key reconciliation, and message signing and verification.

4.1. System initialization phase

Following are the processes by which the TA generates the public parameters of the system.

- The system is set up with an elliptic curve $E : y^2 = x^3 + ax + b \bmod p$, where $a, b \in Z_q^*$ with a condition $\Delta = 4a^3 + 27b^2 \neq 0$ and p is a large prime number. For 80-bit security, we use the recommended domain parameters in [43] of the 160-bit elliptic curve "secp160k1", see Table 3.
- Using the base point g, the TA creates the cyclic additive group $\mathbb{G}$ of order q comprising all the points on E and the point of infinity $\mathcal{O}$.
- The TA selects its own private key $Sk_{TA} \in Z_q^*$ and computes its associated public key $Pk_{TA} = Sk_{TA}.g$.
- The TA selects a unique private key for all the RTAs $Sk_{RTA} \in Z_q^*$ and computes its associated public key $Pk_{RTA} = Sk_{RTA}.g$.
- The hash function $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{N_1}$.

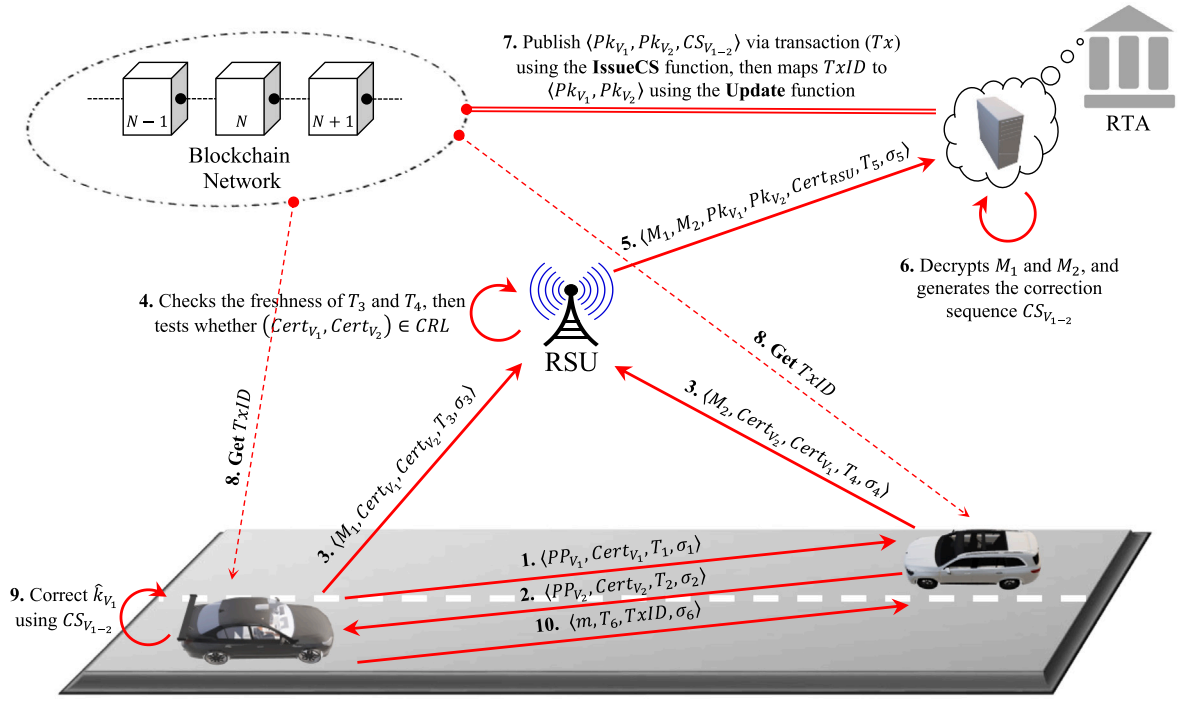


Fig. 4. The proposed blockchain-based authentication model for VANETs.

Table 3
The recommended domain parameters of the 160-bit elliptic curve "secp160k1" in the hexadecimal form [43].

Par.	Recommended value
a	00000000 00000000 00000000 00000000 00000000
b	00000000 00000000 00000000 00000000 00000007
p	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFAC73
g	04 3B4C382C E37AA192 A4019E76 3036F4F5 DD4D7EBB 938CF935 318FDCED 6BC28286 531733C3 F03C4FEE
q	01 00000000 00000000 0001B8FA 16DFAB9A CA16B6B3

- The TA deploys the SC on behalf of the RTA (i.e., using the RTA's address), then obtains the SC's unique identity/address $SCID$. The network terminals use the obtained $SCID$ to call the **Get** function from the deployed SC to attain their CS.
- Finally, the public parameters of the scheme are $PPs = \langle a, b, p, q, g, Pk_{TA}, H_1, SCID\rangle$.

4.2. Registration phase

The TA is responsible for registering all the terminals before being part of the network by doing the steps below.

- For registering a vehicle V_i, the TA checks V_i's real identity RID_{V_i}, picks up at random V_i's private key $Sk_{V_i} \in Z_q^*$, then computes its associated public key $Pk_{V_i} = Sk_{V_i}.g$. At last, TA creates $Cert_{V_i} = \langle Pk_{V_i}, T_R, \sigma_{TA}\rangle$ in which $\sigma_{TA} = \text{sign}_{Sk_{TA}}(Pk_{V_i} \parallel T_R)$ and T_R is the expiry date of the certificate. This is also done for all registered RTAs and RSUs in the network.
- As a final step, the TA preloads $\langle PPs, Pk_{RTA}, Sk_{V_i}, Cert_{V_i}\rangle$ onto the registered V_i, $\langle PPs, Pk_{RTA}, Sk_{RSU_j}, Cert_{RSU_j}\rangle$ onto the registered RSU_j, and $\langle PPs, Sk_{RTA}, Cert_{RTA}\rangle$ onto the registered RTA. Note that, only the TA has the link between the RID_{V_i} and $Cert_{V_i}$ to reveal V_i's real identity in case of malicious activity.

Fig. 5 shows the top-level description flowchart of the proposed scheme's phases following the registration phase.

4.3. Initial verification and channel probing phase

In this phase, both terminals exchange authenticated probing packets PP_{V_i} along with digital certificates used for establishing a shared key and mutual authentication. This phase comprises the following steps:

- *Step 1*: During the first transmission slot, V_1 sends V_2 a communication request in the form of $\langle PP_{V_1}, Cert_{V_1}, T_1, \sigma_1\rangle$, where $\sigma_1 = \text{sign}_{Sk_{V_1}}(PP_{V_1} \parallel Cert_{V_1} \parallel T_1)$ generated at timestamp T_1.
- *Step 2.1*: V_2 replies by sending the tuple $\langle PP_{V_2}, Cert_{V_2}, T_2, \sigma_2\rangle$ to V_1, for $T_2 - T_1 \le T_c$, where $\sigma_2 = \text{Sign}_{Sk_{V_2}}(PP_{V_2} \parallel Cert_{V_2} \parallel T_2)$ generated at timestamp T_2. After that, both vehicles check if $(Cert_{V_1}, Cert_{V_2}) \in$ CRL. If not, they check the freshness of the received timestamp by finding out if $T_r - T_i \le T_\Delta$ holds or not, defending against replay attacks. Then, both verify the received signatures as $\text{verf}_{Pk_{V_{1(2)}}}(\sigma_{1(2)})$.
- *Step 2.2*: Based on the short-term channel reciprocity, both vehicles obtain their channel phase response estimates, specified in (5), and quantize them using (6) to get the bit streams $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$ at the side of V_1 and V_2, respectively. However, $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$ hold some mismatched bits resulting from the channel non-reciprocity components.
- *Step 3*: Accordingly, both vehicles encrypt their secret keys to get $M_{1(2)} = \text{Enc}_{k_{V_{1(2)}-RTA}}(\hat{k}_{V_{1(2)}})$ in which $k_{V_{1(2)}-RTA} = Sk_{V_{1(2)}}.Pk_{RTA}$ and send it to the RSU in the form of

$$\begin{aligned} &V_1 \rightarrow RSU : \langle M_1, Cert_{V_1}, Cert_{V_2}, T_3, \sigma_3\rangle, \\ &V_2 \rightarrow RSU : \langle M_2, Cert_{V_2}, Cert_{V_1}, T_4, \sigma_4\rangle \end{aligned} \tag{9}$$

  where $\sigma_{3(4)} = \text{Sign}_{Sk_{V_{1(2)}}}(M_{1(2)} \parallel Cert_{V_{1(2)}} \parallel Cert_{V_{2(1)}} \parallel T_{3(4)})$ generated at $T_{3(4)}$ timestamp.
- *Step 4*: The RSU in turn checks the freshness of the received timestamp $T_{3(4)}$, checks if $(Cert_{V_1}, Cert_{V_2}) \in$ CRL, and then verifies the received signatures $\text{verf}_{Pk_{V_{1(2)}}}(\sigma_{3(4)})$.
- *Step 5*: The RSU forward the encrypted secret keys to the RTA as

$$\langle M_1, M_2, Pk_{V_1}, Pk_{V_2}, Cert_{RSU}, T_5, \sigma_5\rangle \tag{10}$$

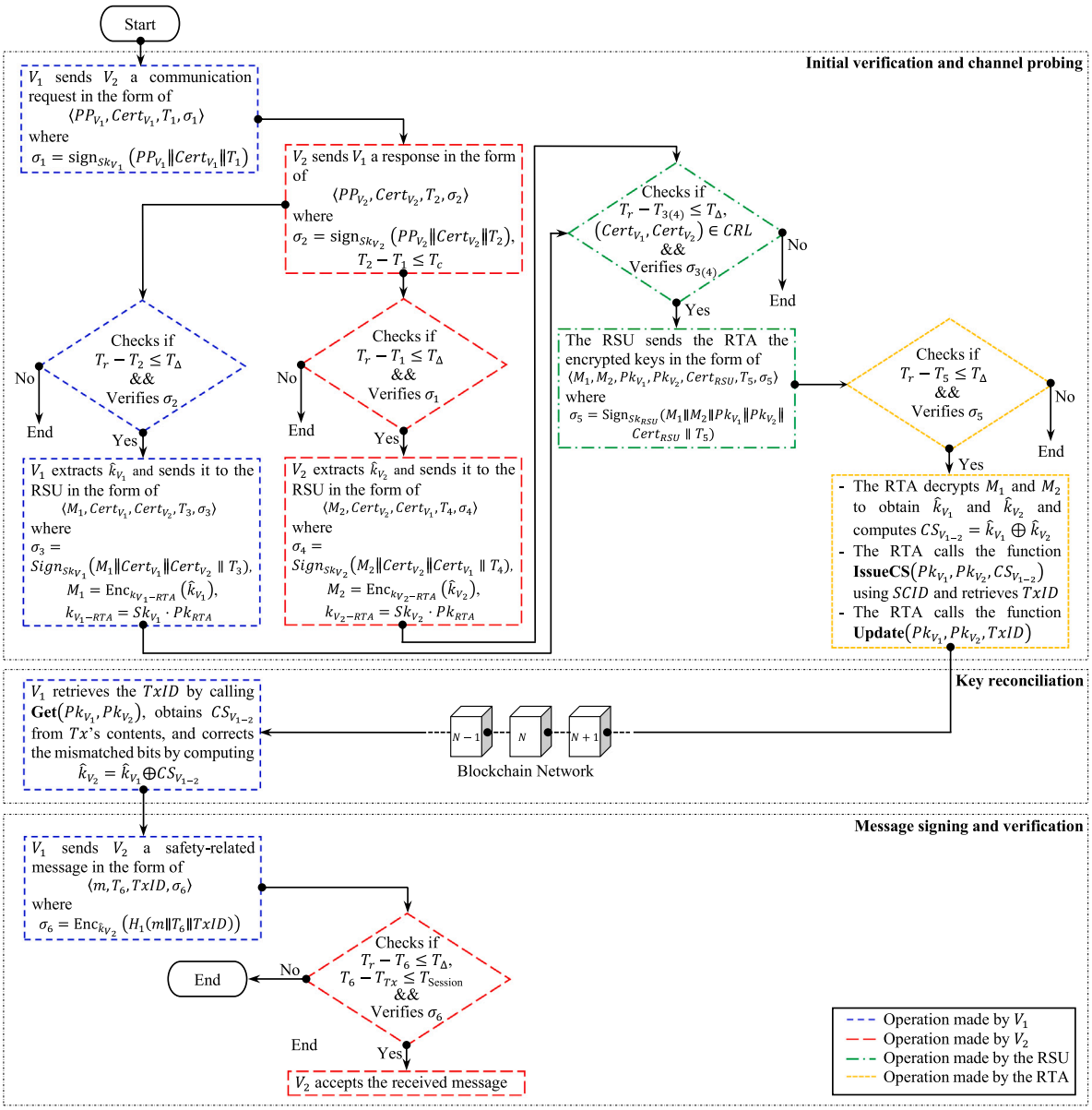


Fig. 5. The top-level description flowchart of the proposed scheme.

where $\sigma_5 = \text{Sign}_{Sk_{RSU}}(M_1 \parallel M_2 \parallel Pk_{V_1} \parallel Pk_{V_2} \parallel Cert_{RSU} \parallel T_5)$ generated at T_5 timestamp.

- *Step 6*: The RTA checks T_5, verifies the received signature $\text{verf}_{Pk_{RSU}}(\sigma_5)$, then decrypts M_1 and M_2 to get $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$, respectively, as $\hat{k}_{V_{1(2)}} = \text{Dec}_{k_{V_{1(2)}-RTA}}(M_{1(2)})$ in which $k_{V_{1(2)}-RTA} = Sk_{RTA}.Pk_{V_{1(2)}}$.
- *Step 7*: Accordingly, if the RTA finds a sufficient matching percentage between $\hat{k}_{V_1}$ and $\hat{k}_{V_2}$, it computes the correction sequence $CS_{V_{1-2}} = \hat{k}_{V_1} \bigoplus \hat{k}_{V_2}$, and then records $CS_{V_{1-2}}$ into the blockchain by calling **IssueCS**$(Pk_{V_1}, Pk_{V_2}, CS_{V_{1-2}})$ using $SCID$. Once the miners chain the transaction into the blockchain at time T_{Tx} and the RTA obtains the transaction identity $TxID$, it maps the pair of public keys (Pk_{V_1}, Pk_{V_2}) to $TxID$ by calling **Update**$(Pk_{V_1}, Pk_{V_2}, TxID)$ in the SC using the $SCID$.

4.4. Key reconciliation

In this phase, V_1 reconciles the mismatched bits in $\hat{k}_{V_1}$ by performing the following steps.

- V_1 obtains $TxID$ by calling **Get**(Pk_{V_1}, Pk_{V_2}) using $SCID$ to get $CS_{V_{1-2}}$ from the blockchain and agrees on a shard key with V_2 by computing $\hat{k}_{V_2} = \hat{k}_{V_1} \bigoplus CS_{V_{1-2}}$.

4.5. Message signing and verification phase

In this phase, V_1 signs a time-stamped safety-related message m using symmetric key cryptography and sends it to V_2. Using the key obtained, V_2 verifies the received signature and accepts the received message.

- *Step 1*: V_1 sends V_2 the tuple $\langle m, T_6, TxID, \sigma_6 \rangle$, where $\sigma_6 = \text{Enc}_{\hat{k}_{V_2}}(H_1(m \parallel T_6 \parallel TxID))$ generated at T_6 timestamp.
- *Step 2*: V_2 checks the freshness of T_6 timestamp, invokes the $TxID$ data from the blockchain to verify the session's continuity by checking if $T_6 - T_{Tx} \le T_{Session}$ holds or not, then decrypts σ_6 to verify the integrity of the attached data by testing whether $\text{Dec}_{\hat{k}_{V_2}}(\sigma_6) \stackrel{?}{=} H_1(m \parallel T_6 \parallel TxID)$. If true, the message is accepted. Otherwise, it will be discarded.

5. Security proofs and analysis

This section proves the correctness of the BCSKE scheme using BAN-logic, analyzes its security strength and proves its robustness using the AVISPA simulation tool.

5.1. BAN-logic security proof

The BAN-logic is a proof of correctness technique used to verify the validity of the authentication scheme [44]. By using the BAN-logic analysis, we show that the proposed BCSKE scheme provides successful key agreement and authentication processes. Table 4 shows the BCSKE scheme used notations and their corresponding BAN-logic symbols.

(1) *Notations*: The following notations are used for the BAN-logic security proof:

(a) $A \mid\equiv X$: A believes X and accepts it as true.
(b) $A \lhd X$: A sees X, indicating A received a message containing X.
(c) $A \mid\sim X$: X has once transmitted and believed by A at one time.
(d) $A \mid\Longrightarrow X$: A controls X and has jurisdiction over it.
(e) $A \overset{k}{\longleftrightarrow} B$: A and B use k as a shared key for communication.
(f) $A \overset{k}{\longrightarrow} B$: k represents the public key of A.
(g) $\{X\}_k$: X is encrypted using the shared key k.
(h) $\#(X)$: X is a fresh message.

(2) *Rules*: A set of beliefs can be generated by manipulating the protocol according to the following rules.

(a) *Message meaning rule* (MMR):

i. For a symmetric shared key:

$$\frac{A \mid\equiv (A \overset{K}{\longleftrightarrow} B), A \lhd \{X\}_K}{A \mid\equiv (B \mid\sim X)} \tag{11}$$

ii. For a public key:

$$\frac{A \mid\equiv (B \overset{K}{\longrightarrow} A), A \lhd \{X\}_{k^{-1}}}{A \mid\equiv (B \mid\sim X)} \tag{12}$$

(b) *Nonce verification rule* (NVR):

$$\frac{A \mid\equiv \#(X), A \mid\equiv (B \mid\sim X)}{A \mid\equiv (B \mid\equiv X)} \tag{13}$$

(c) *Jurisdiction rule* (JR):

$$\frac{A \mid\equiv (B \Longrightarrow X), A \mid\equiv (B \mid\equiv X)}{A \mid\equiv X} \tag{14}$$

(d) *Freshness rule* (FR):

$$\frac{A \mid\equiv \#(X)}{A \mid\equiv \#(X, Y)} \tag{15}$$

(3) *Goals*: In BAN-logic, we aim to prove the correctness of the proposed scheme by satisfying the following goals.

(a) *Goal 1*: $V_1 \mid\equiv (V_1 \overset{\hat{k}_{V_2}}{\longleftrightarrow} V_2)$
(b) *Goal 2*: $V_2 \mid\equiv (V_1 \overset{\hat{k}_{V_2}}{\longleftrightarrow} V_2)$
(c) *Goal 3*: $V_2 \mid\equiv (V_1 \mid\sim m)$

(4) *Idealized forms*: Following are the outlines of the idealized messaging forms for the proposed protocol.

(a) $Msg_1 : V_1 \rightarrow V_2 : \{PP_{V_1}, Cert_{V_1}, T_1\}_{k_{V_1}^{-1}}$, where $Cert_{V_1} = \{k_{V_1}, T_R\}_{k_{TA}^{-1}}$.

Table 4
BAN-logic symbols and their equivalent scheme notations.

BAN-logic variable	Scheme notation
$k_{V_i}^{-1}$	Sk_{V_i}
k_{V_i}	Pk_{V_i}
T_R	T_R
$\{k_{V_i}, T_R\}_{k_{TA}^{-1}}$	$Cert_{V_i}$
$\{x\}_{k^{-1}}$	σ_i
T_i	T_i
$\hat{k}_{V_i}$	$\hat{k}_{V_i}$
$k_{RSU_j}^{-1}$	Sk_{RSU_j}
k_{RSU_j}	Pk_{RSU_j}
k_{RTA}^{-1}	Sk_{RTA}
k_{RTA}	Pk_{RTA}
k_{TA}^{-1}	Sk_{TA}
k_{TA}	Pk_{TA}
,	$\|\|$

(b) $Msg_2 : V_2 \rightarrow V_1 : \{PP_{V_2}, Cert_{V_2}, T_2\}_{k_{V_2}^{-1}}$, where $Cert_{V_2} = \{k_{V_2}, T_R\}_{k_{TA}^{-1}}$.
(c) $Msg_3 : V_1 \rightarrow$ RSU: $\{M_1, Cert_{V_1}, Cert_{V_2}, T_3\}_{k_{V_1}^{-1}}$, where $M_1 = \{\hat{k}_{V_1}\}_{k_{V_1-RTA}}$ and $k_{V_1-RTA} = k_{V_1}^{-1}$. $k_{RTA} = k_{RTA}^{-1}.k_{V_1}$ using Diffie–Hellman protocol.
(d) $Msg_4 : V_2 \rightarrow$ RSU: $\{M_2, Cert_{V_2}, Cert_{V_1}, T_4\}_{k_{V_2}^{-1}}$, where $M_2 = \{\hat{k}_{V_2}\}_{k_{V_2-RTA}}$ and $k_{V_2-RTA} = k_{V_2}^{-1}$. $k_{RTA} = k_{RTA}^{-1}.k_{V_2}$ using Diffie–Hellman protocol.
(e) $Msg_5 : RSU \rightarrow$ RTA: $\{M_1, M_2, k_{V_1}, k_{V_2}, T_5, Cert_{RSU}\}_{k_{RSU}^{-1}}$, where $Cert_{RSU} = \{k_{RSU}, T_R\}_{k_{TA}^{-1}}$.
(f) $Msg_6 : V_1 \rightarrow V_2 : \{m, T_6, TxID\}_{\hat{k}_{V_2}}$.

(5) *Assumptions*: Following are the basic assumptions that underlie the BAN logic security proof.

(a) A_1: $V_2 \mid\equiv \#(T_1)$
(b) A_2: $V_1 \mid\equiv \#(T_2)$
(c) A_3: $RSU \mid\equiv \#(T_3)$
(d) A_4: $RSU \mid\equiv \#(T_4)$
(e) A_5: $RTA \mid\equiv \#(T_5)$
(f) A_6: $V_2 \mid\equiv \#(T_6)$
(g) A_7: $RTA \mid\equiv RTA \overset{k_{V_1-RTA}}{\longleftrightarrow} V_1$
(h) A_8: $RTA \mid\equiv RTA \overset{k_{V_2-RTA}}{\longleftrightarrow} V_2$
(i) A_9: $V_2 \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} V_2)$
(j) A_{10}: $\dfrac{V_2 \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} V_2), V_2 \lhd \{k_{V_1}, T_R\}_{k_{TA}^{-1}}}{V_2 \mid\equiv (V_1 \overset{k_{V_1}}{\longrightarrow} V_2)}$
(k) A_{11}: $V_1 \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} V_1)$
(l) A_{12}: $\dfrac{V_1 \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} V_1), V_1 \lhd \{k_{V_2}, T_R\}_{k_{TA}^{-1}}}{V_1 \mid\equiv (V_2 \overset{k_{V_2}}{\longrightarrow} V_1)}$
(m) A_{13}: $RSU \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} RSU)$
(n) A_{14}: $\dfrac{RSU \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} RSU), RSU \lhd \{k_{V_1}, T_R\}_{k_{TA}^{-1}}}{RSU \mid\equiv (V_1 \overset{k_{V_1}}{\longrightarrow} RSU)}$
(o) A_{15}: $\dfrac{RSU \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} RSU), RSU \lhd \{k_{V_2}, T_R\}_{k_{TA}^{-1}}}{RSU \mid\equiv (V_2 \overset{k_{V_2}}{\longrightarrow} RSU)}$
(p) A_{16}: $RTA \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} RTA)$
(q) A_{17}: $\dfrac{RTA \mid\equiv (TA \overset{K_{TA}}{\longrightarrow} RTA), RTA \lhd \{k_{RSU}, T_R\}_{k_{TA}^{-1}}}{RTA \mid\equiv (RSU \overset{k_{RSU}}{\longrightarrow} RTA)}$

(6) *Implementation*: Following is the BAN-logic security proof to the proposed protocol.

- *Step 1*: V_2 receives Msg_1 from V_1.
- *Step 2*: Applying A_9 and $Cert_{V_1}$ from Msg_1 into A_{10}, then the result is $R_1 : V_2 \mid\equiv (V_1 \overset{k_{V_1}}{\longrightarrow} V_2)$. Substituting R_1 and Msg_1 into the MMR in the public key form, then $R_2 : V_2 \mid \equiv (V_1 \mid\sim Msg_1)$. Applying A_1 and Msg_1 into the FR, then $R_3 : V_2 \mid\equiv \#(Msg_1)$. By combining R_2 and R_3 into the NVR, then $R_4 : V_2 \mid\equiv (V_1 \mid\equiv Msg_1)$.
- *Step 3*: V_1 receives Msg_2 from V_2.
- *Step 4*: Applying A_{11} and $Cert_{V_2}$ from Msg_2 into A_{12}, then $R_5 : V_1 \mid\equiv (V_2 \overset{k_{V_2}}{\longrightarrow} V_1)$. Substituting R_5 and Msg_2 into the MMR in the public key form, then $R_6 : V_1 \mid \equiv (V_2 \mid\sim Msg_2)$. Applying A_2 and Msg_2 into the FR, then $R_6 : V_1 \mid\equiv \#(Msg_2)$. By combining R_6 and R_7 into the NVR, then $R_8 : V_1 \mid\equiv (V_2 \mid\equiv Msg_2)$.
- *Step 5*: RSU receives Msg_3 from V_1.
- *Step 6*: Applying A_{13} and $Cert_{V_1}$ from Msg_3 into A_{14}, then $R_9 : RSU \mid\equiv (V_1 \overset{k_{V_1}}{\longrightarrow} RSU)$. Substituting R_9 and Msg_3 into the MMR in the public key form, then $R_{10} : RSU \mid \equiv (V_1 \mid\sim Msg_3)$. Applying A_3 and Msg_3 into the FR, then $R_{11} : RSU \mid\equiv \#(Msg_3)$. By combining R_{10} and R_{11} into the NVR, then $R_{12} : RSU \mid\equiv (V_1 \mid\equiv Msg_3)$.
- *Step 7*: RSU receives Msg_4 from V_2.
- *Step 8*: Applying A_{13} and $Cert_{V_2}$ from Msg_4 into A_{15}, then $R_{13} : RSU \mid\equiv (V_2 \overset{k_{V_2}}{\longrightarrow} RSU)$. Substituting R_{13} and Msg_4 into the MMR in the public key form, then $R_{14} : RSU \mid \equiv (V_2 \mid\sim Msg_4)$. Applying A_4 and Msg_4 into the FR, then $R_{15} : RSU \mid\equiv \#(Msg_4)$. By combining R_{14} and R_{15} into the NVR, then $R_{16} : RSU \mid\equiv (V_2 \mid\equiv Msg_4)$.
- *Step 9*: RTA receives Msg_5 from RSU.
- *Step 10*: Applying A_{16} and $Cert_{RSU}$ from Msg_5 into A_{17}, then $R_{17} : RTA \mid\equiv (RSU \overset{k_{RSU}}{\longrightarrow} RTA)$. Substituting R_{17} and Msg_5 into the MMR in the public key form, then $R_{18} : RTA \mid \equiv (RSU \mid\sim Msg_5)$. Applying A_5 and Msg_5 into the FR, then $R_{19} : RTA \mid\equiv \#(Msg_5)$. By combining R_{18} and R_{19} into the NVR, then $R_{20} : RTA \mid\equiv (RSU \mid\equiv Msg_5)$.
- *Step 11*: Applying A_7 and M_1 from Msg_5 into the MMR in the shared key form, then $R_{21} : RTA \mid\equiv (V_1 \mid\sim M_1)$. Similarly, by applying A_8 and M_2 from Msg_5 into MMR in a symmetric shared key form, then $R_{22} : RTA \mid\equiv (V_2 \mid\sim M_2)$. Based on R_{21} and R_{22}, the RTA can calculate $CS_{V_{1-2}} = \hat{k}_{V_1} \bigoplus \hat{k}_{V_2}$ and record it into the blockchain. Accordingly, V_1 retrieves the $CS_{V_{1-2}}$ from the blockchain using $TxID$ and agrees with V_2 on $\hat{k}_{V_2}$. Now, $V_1 \mid\equiv (V_1 \overset{\hat{k}_{V_2}}{\longleftrightarrow} V_2)$ (***Goal 1***).
- *Step 12*: V_2 receives Msg_6 from V_1.
- *Step 13*: Once $V_2 \lhd TxID$ in the blockchain, then $V_2 \mid\equiv (V_2 \overset{\hat{k}_{V_2}}{\longleftrightarrow} V_1)$ (***Goal 2***). Thus, by applying ***Goal 2*** and Msg_6 into the MMR in the shared key form, then $V_2 \mid\equiv (V_1 \mid\sim Msg_6)$ (***Goal 3***).

5.2. Security analysis

Throughout this subsection, we discuss the security requirements fulfilled through our methodology, which are primarily determined by the digital signatures and blockchain system adopted.

1. *Message authentication*: The proposed scheme's security strength mainly depends on the infeasibility of solving the elliptic curve discrete logarithm problem (ECDLP) for the first transmission slot and the sufficient security level provided by the symmetric key-based cryptography with a key length of order $|\hat{k}_{V_2}| \simeq 128$, 192, or 256 bits during subsequent transmission slots. Furthermore, the certificate $Cert_{V_i}$ signed by the TA allows the recipient to authenticate the sender's public key Pk_{V_i}. Hence, the recipient is able to authenticate the received message by verifying the received signatures $\mathrm{verf}_{Pk_{V_i}}(\sigma_i)$ and $\mathrm{Dec}_{\hat{k}_{V_i}}(\sigma_i) \overset{?}{=} H_1(m \parallel T_i \parallel TxID)$ for first and subsequent transmission slots, respectively.
2. *Conditional privacy preservation*: Since all the transmitted messages have no information about V_i's real identity RID_{V_i}, no network terminal is able to expose RID_{V_i} except for TA, as it is the only terminal that stores the link between RID_{V_i} and V_i's issued long-term digital certificate $Cert_{V_i}$. Hence, preserving privacy under certain conditions.
3. *Unlinkability*: Since the secret key extraction process depends on the reciprocal features and the spatially and temporally correlated wireless channel responses within T_c, the adversary cannot establish a rational relationship between the extracted keys from different sessions, supporting forward and backward secrecy. In addition, the dynamically updated $TxID$ between vehicles prevents adversaries from linking messages from different sessions.
4. *Resistance to birthday collisions*: Ethereum's consensus mechanism depends on the proof of work (Ethereum 1.0) and proof of stake (Ethereum 2.0). This consensus mechanism helps prevent forking; thus, the likelihood of a block's birthday colliding is effectively reduced.
5. *Resistance to Hijacking*: All Ethereum transactions are signed using the digital signatures of the elliptic curve (secp256k1). The ECDSA's security ensures that no probabilistic polynomial time adversary can alter the signature of a transaction message, resisting this type of attack.
6. *Resistance to active attacks*: In order for the proposed scheme to be effective, it must be immune to the following types of active attacks.
   (a) *Resistance to modification*: The design of the BCSKE scheme ensures message integrity since V_j can detect modification attempts in the received message m from V_i by checking if $\mathrm{Dec}_{\hat{k}_{V_i}}(\sigma_i) \overset{?}{=} H_1(m \parallel T_i \parallel TxID)$, where $\hat{k}_{V_i}$ is the extracted shared key based on the unpredictable channel randomness between V_i and V_j.
   (b) *Resistance to replay*: The freshly extracted shared key $\hat{k}_{V_i}$ between V_i and V_j in each session allows for avoiding replaying attacks from different sessions. In addition, the attached timestamp T_i helps the recipient to check the freshness of the received message during the same session interval. Hence, resisting such attacks.
   (c) *Resistance to impersonation*: To impersonate a legitimate vehicle V_i, the adversary needs to generate a valid signature at the first transmission slot to extract a secret shared key used for subsequent transmissions. In this sense, the adversary must deduce V_i's private key $Sk_{V_i} \in Z_q^*$ from $Pk_{V_i} = Sk_{V_i}.g$ under the difficulty of solving the ECDLP. Hence, protecting against impersonation attacks.

5.3. Security proof based on AVISPA simulation

In this subsection, the "automated validation of internet security protocols and applications" (AVISPA) tool is used to analyze the security robustness of the BCSKE scheme.

1. *Preliminaries*: In [45], Armando et al. developed the AVISPA toolkit, which is a widely used security protocol animator to validate and evaluate the security aspect of applications and internet security protocols. In AVISPA, the high-level protocol specification language (HLPSL) specifies the role played by each network terminal referred to by the agent, which verifies the

security features regarding authentication and data secrecy of the exchanged messages between different agents in the presence of an intruder. Security properties are predefined in a separate section called "goals", based on which the security protocol is classified as SAFE or UNSAFE. As part of AVISPA's toolset, the HLPSL2IF translator is utilized to translate the HLPSL code into the intermediate format (IF), which is integrally crucial for offering and serving adequate input to the various back-ends of the tool. There are four back-ends provided by AVISPA: TA4SP: tree automata-based on automatic approximations for analysis of security protocol, SATMC: SAT-based model checker, OFMC: on-the-fly model checker, and CL-AtSe: constraint logic-based attack searcher. In this paper, the simulation result of the BCSKE scheme is determined using the CL-AtSe back-end, which determines the protocol's resistance to man-in-the-middle (MITM) and replay attacks. Table 5 presents the BCSKE scheme used notations and their associated HLPSL scripting symbols.

2. *Specifications for simulation*: As a first step, we must specify the security goals for the BCSKE simulation, which include the secrecy of the extracted keys K1 and K2 between different agents referred to by sec_1, sec_2, sec_3, and sec_4, along with authenticating the broadcasted messages by the intended agent described by auth_1, auth_2, auth_3, auth_4, auth_5, auth_6, and auth_7. In the simulation, there are four agents' roles role_V1, role_V2, role_RSU, and role_RTA played by V1, V2, RSU, and RTA, respectively. During the role session, all the agents' declarations are defined, and in the role environment, all the variables and functions associated with different agents are denoted. Fig. 6 shows the protocol simulation in the form of transitions between different agents in the BCSKE scheme. A full explanation of these roles is presented in the Appendix in the form of HLPSL codes. Note that /\ means a conjunction between two operations.

Code 1 in the Appendix shows the role played by V1 in the network. The knowledge of V1 includes all the protocol's agents (V1, V2, RSU, and RTA), their public keys (KV1, KV2, KRSU, and KRTA), TA's public key KTA, the symmetric key KV1Rta between RTA and V1 (equals $Sk_{V_1}.Pk_{RTA}$ using the Diffie–Hellman key exchanging protocol), and the send (SND) and receive (RCV) Dolev-Yao (dy) channels. The local variables part defines the role's initial state (State:= 0), the certificate expiry date TR, the timestamps (T1, T2, T4, and T6), the probing packets of V1 and V2 (PPV1 and PPV2), the correction sequence CS, the extracted keys by V1 and V2 (K1 and K2), and the message M. The following are the three transitions that describe the role of V1.

- For State = 0, and if V1 receives the start signal "RCV(start)" to execute the protocol, then the current state is increased by 1 (State':= 1) and V1 sends the communication request containing the probing packet PPV1, a fresh timestamp T1', V1's certificate, and the message signature signed by V1's private key inv(KV1). Note that {x}_inv(y) represents the signature of the contents x using the agent's private key inv(y). Finally, V1 expects that V2 authenticates PPV1 through a process named "auth_1".
- For State = 1, and if V1 receives a message from V2 containing the probing packet PPV2, a fresh timestamp T2', V2's certificate, and the message signature signed by V2's private key inv(KV2), then the current state is increased by 1 (State':= 2) and V1 sends a message containing the encrypted key {K1'}_KV1Rta, V1's certificate, V2's certificate, a fresh timestamp T4', and the message signature signed by V1's private key inv(KV1). Finally, V1 verifies the received PPV2 from V2 through a process named "auth_2", believes in the secrecy of the transmitted K1' to the RSU through a process named "sec_1", and expects that RSU authenticates {K1'} _KV1Rta through a process named "auth_4".
- In the 3rd transition, we refer to the obtained correction sequence from the blockchain as a received message from the RTA containing KV1, KV2, and CS' signed by the RTA's private key as {KV1.KV2.CS'} _inv(KRTA). For State = 2, and if V1 receives a signed CS' from the RTA, then V1 reconciles the mismatched bits by computing K2':= xor(CS',K1) and securely sends a message to V2 containing the safety-related message M, a fresh timestamp T6', the CS', and the message signature signed by the symmetric key K2'. Finally, V1 verifies the received CS' from the RTA through a process named "auth_6", and hopes that M will be authenticated by V2 through a process named "auth_7".

Table 5
AVISPA symbols and their equivalent scheme notations.

HLPSL variable	Scheme notation
inv(KVi)	Sk_{V_i}
KVi	Pk_{V_i}
TR	T_R
KVi.TR.{KVi.TR}_inv(KTA)	$Cert_{V_i}$
{x}_inv(k)	σ_i
Ti	T_i
Ki	$\hat{k}_{V_i}$
inv(KRSU)	Sk_{RSU_j}
KRSU	Pk_{RSU_j}
inv(KRTA)	Sk_{RTA}
KRTA	Pk_{RTA}
inv(KTA)	Sk_{TA}
KTA	Pk_{TA}
CS	$CS_{V_{1-2}}$
KV1.KV2.CS.{KV1.KV2.CS}_inv(KRTA)	$TxID$
.	‖

Code 2 in the Appendix shows the role played by V2 in the network. The knowledge of V2 includes all the protocol's agents (V2, V1, RSU, and RTA), their public keys (KV2, KV1, KRSU, and KRTA), TA's public key KTA, the symmetric key KV2Rta between V2 and RTA (equals $Sk_{V_2}.Pk_{RTA}$ using the Diffie–Hellman key exchanging protocol), and the SND/RCV channels. The local variables part defines the role's initial state (State:= 0), the certificate expiry date TR, the timestamps (T1, T2, T3, and T6), the probing packets of V1 and V2 (PPV1 and PPV2), the correction sequence CS, the extracted keys by V1 and V2 (K1 and K2), and the message M. The following are the two transitions that describe the role of V2.

- For State = 0, and if V2 receives the communication request from V1, then the current state is increased by 1 (State':=1) and V2 sends

  - a reply message to V1 containing the probing packet PPV2, a fresh timestamp T2', V2's certificate, and the message signature signed by V2's private key inv(KV2). Finally, V2 verifies the received PPV1 from V1 through a process named "auth_1" and expects that V1 authenticates PPV2 through a process named "auth_2".
  - a message to the RSU containing the encrypted key {K2'}_KV2Rta using the symmetric key KV2Rta, V1's certificate, V2's certificate, a fresh timestamp T3', and the message signature signed by V2's private key inv(KV2). Finally, V2 believes in the secrecy of the transmitted K2' to the RSU through a process named "sec_2" and expects that RSU authenticates {K2'}_KV2Rta through a process named "auth_3".

- For State = 1, and if V2 receives a message from V1 containing the safety-related message M, a fresh timestamp T6', the CS', and message signature signed by the symmetric key K2, then the current state is increased by 1 (State':= 2). Finally, V2 verifies the received M from V1 through a process named "auth_7".

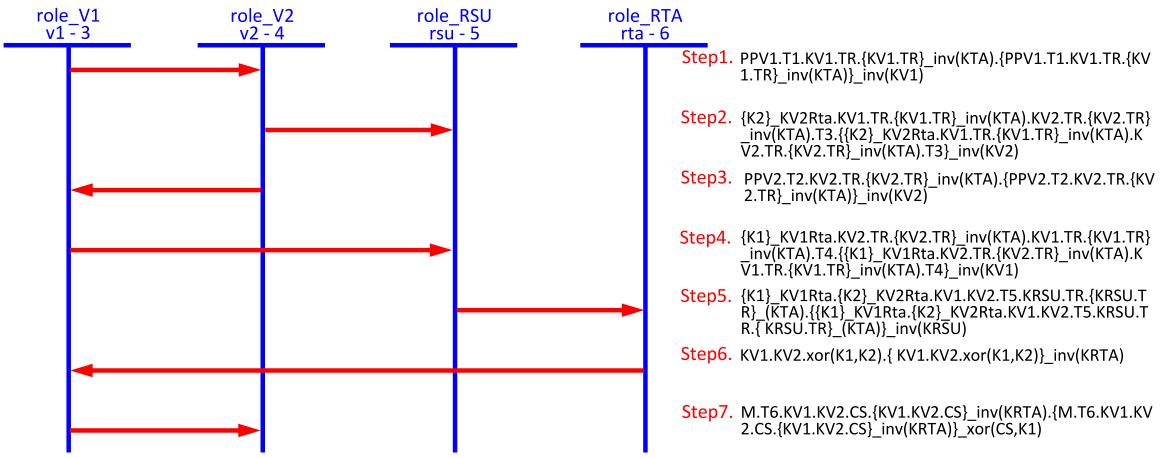


Fig. 6. AVISPA protocol simulation.

Code 3 in the Appendix shows the role played by the RSU in the network. The knowledge of the RSU includes all the protocol's agents (RSU, V1, V2, and RTA), their public keys (KRSU, KV1, KV2, and KRTA), TA's public key KTA, the SND/RCV channels. The local variables part defines the role's initial state (State:= 0), the certificate expiry date TR, the timestamps (T3, T4, and T5), and the symmetric keys (K1, K2, KV1Rta, and KV2Rta). The following are the two transitions that describe the role of the RSU.

- For State = 0, and if the RSU receives a message from V2 containing the encrypted key {K2'}_KV2Rta using the symmetric key KV2Rta, V1's certificate, V2's certificate, a fresh timestamp T3', and the message signature signed by V2's private key inv(KV2), then the current state is increased by 1 (State':= 1). Finally, the RSU verifies the received {K2'} _KV2Rta' from V2 through a process named "auth_3".
- For State = 1, and if the RSU receives a message from V1 containing the encrypted key {K1'}_KV1Rta, V1's certificate, V2's certificate, a fresh timestamp T4', and the message signature signed by V1's private key inv(KV1), then the current state is increased by 1 (State':= 2) and the RSU sends a message to the RTA containing the encrypted keys {K1'}_KV1Rta' and {K2'}_KV2Rta', V1's certificate, V2's certificate, a fresh timestamp T5', and the message signature signed by the RSU's private key. Finally, the RSU verifies the received {K1'} _KV1Rta' from V1 through a process named "auth_4", believes in the secrecy of the transmitted K1' and K2' to the RTA through a process named "sec_3" and "sec_4", respectively, and expects that the RTA authenticates {K1'}_KV1Rta and {K2'}_KV2Rta through a process named "auth_5".

Code 4 in the Appendix shows the role played by the RTA in the network. The knowledge of the RTA includes all the protocol's agents (V2, V1, RSU, and RTA), their public keys (KV2, KV1, KRSU, and KRTA), TA's public key KTA, the symmetric key KV1Rta and KV2Rta, the SND/RCV channels. The local variables part defines the role's initial state (State:= 0), the certificate expiry date TR, the timestamps T5, and the symmetric keys K1 and K2. There is a single transition played by the RTA denoted by

- For State = 0, and if the RTA receives a message from the RSU containing the encrypted keys, then the current state is increased by 1 (State':= 1), the RTA computes CS':= xor(K1', K2') and sends a message to V1 containing (KV1, KV2, CS') and the message signature signed by the RTA's private key inv(KRTA). Finally, the RTA verifies the received {K1'} _KV1Rta' and {K1'} _KV1Rta' from the RSU through a process named "auth_5", and expects that V1 authenticates CS' through a process named "auth_6".

As a final substep, **Code 5** in the Appendix defines the protocol variables and the intruder knowledge of all the network agents and their associated public keys. In addition, the same code outlines the protocol goals mentioned above.

3. *Simulation results*: Based on the AVISPA security analysis, Fig. 7 summarizes the simulation result of the specified goals using the Cl-AtSe back-end checker. As can be seen, the CL-AtSe model takes 0.03 s for IF translation. According to the summary, we can conclude that the BCSKE protocol is SAFE from potential MITM and replay attacks.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/BCSKE protocol.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed   : 38 states
 Reachable  : 10 states
 Translation: 0.03 seconds
 Computation: 0.00 seconds
```

Fig. 7. AVISPA simulation result using CL-AtSe.

6. Performance analysis

6.1. Implementation and transaction fees

In this part, we discuss the functionality of the BCSKE scheme by implementing it on the Ethereum main network, i.e., *Ethereum MainNet*. For triggering and deploying the proposed *SC*, we integrated *Ethereum MainNet* and *Remix* 0.25.1 using *MetaMask* (a Microsoft Edge plug-in extension). For all *Metamask* wallets, *Ethereum MainNet* is the default network that is used by developers to develop and examine the actual performance of various types of decentralized applications. Following are the details of the BCSKE implementation.

Table 6
Gas costs for different SC's functions (1 ETH = 1859.89 $).

Function	Gas used (Gwei)	Actual cost (ETH)
Deployer	17.06409157	0.0052635896856822
IssueCS	15.260894856	0.00139812688223244
Update	15.465276944	0.001416851346989
Get	No fees	No fees

1. As a first step, we created three accounts in *MetaMask* corresponding to RTA, V_1, and V_2, denoted by 0xcbB3 012b86b594223E43FB9c50176624F357463b, 0xa30C281D2Cf 6252e7524f38341fb6d1a8b68876B, and 0xA510aEe2869D2A 2608C6D96d454d322BC67d327A, respectively, as shown in Fig. 8(a). Then switched to the RTA's account and charged it from the *Ethereum MainNet* such that the RTA can deploy and interact with the SC's functions, i.e., **Deployer**, **IssueCS**, **Update**, and **Get**. Afterward, we deployed the proposed SC into the blockchain and retrieved its address ($SCID$), denoted by 0xfeA69D128a4C82AffA49a70E95b35774e04738, as shown in Fig. 8(b). An over-view of the SC deployment process is given in Fig. 9, which includes information on transaction fees, gas costs (Ethereum unit of measurement, i.e., ETH and Wei), etc.
2. Following that, we invoked the **IssueCS** function to deploy the correction sequence (CS) of the mismatched bits in a transaction (Tx) and obtain its related address ($TxID$). Next, we mapped the $TxID$ to both of the communicating vehicles' public keys (Pk_{V_1}, Pk_{V_2}) using the **Update** function.
3. Finally, we switched to V_1 to obtain the CS of the transaction (Tx) by invoking the **Get** function to attain $TxID$.

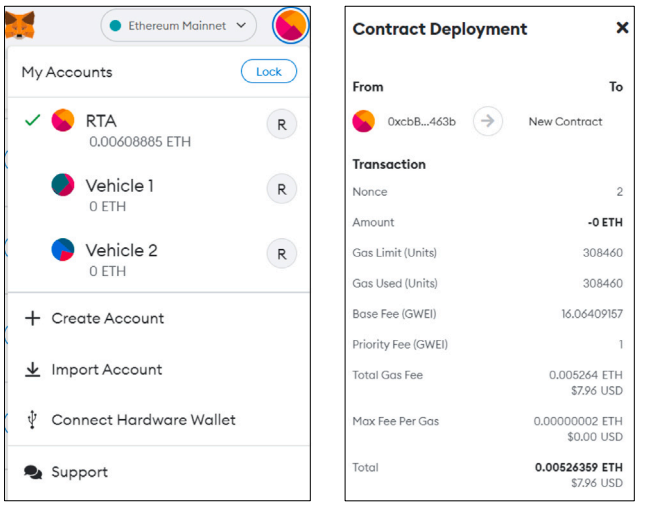
An example of how the network terminals interact with the SC's different functions is explained in a four-step process using the *Remix Virtual Machine* - see Fig. 10, as follows.

- *Step 1*: Using the **Deployer** function, the SC is deployed by the RTA.
- *Step 2*: Assuming Pk_{V_1} = 847932647234870754345, Pk_{V_2} = 234354679832720343455, and CS = 1054342 3726987432 3123, we published the CS via a transaction and retrieved its related address, e.g., $TxID$ = 782353723486592342345 using the **IssueCS** function.
- *Step 3*: The **Update** function maps the obtained $TxID$ to Pk_{V_1} and Pk_{V_2}.
- *Step 4*: The **Get** function is used to call the $TxID$ corresponding to the input public keys Pk_{V_1} and Pk_{V_2}.

In addition, we evaluated the gas costs associated with the SC's functions, as shown in Table 6. As can be seen, the SC deployment is the most costly phase, but it only needs to be performed once. In contrast to all the other functions, the **IssueCS** and **Update** functions are invoked at the beginning of each session. While the **Get** function is invoked by the recipient for every received safety-related message. A comparison of transaction time and fee costs for various blockchains is shown in Table 7 [46–48]. Among its rivals, it is noteworthy that Aleph Zero is the fastest blockchain (~600 ms) with the lowest transaction fees (~0.0003$). In each session, based on Aleph Zero statistics, the communicating vehicles lose from two to six safety-related messages since the broadcasting rate is a message every 100–300 ms. Fortunately, blockchain technology is rapidly developing, which can help mitigate this loss in the future.

6.2. Comparative analysis of computation cost

First, we invoked the same evaluation of the time consumed by different cryptographic operations in [40] - see Table 8. These operations are measured using the MIRACL library [50] running on quad-core



(a) Network terminals. (b) Smart contract deployment.

Fig. 8. Blockchain terminals and the SC deployment process in *MetaMask*.

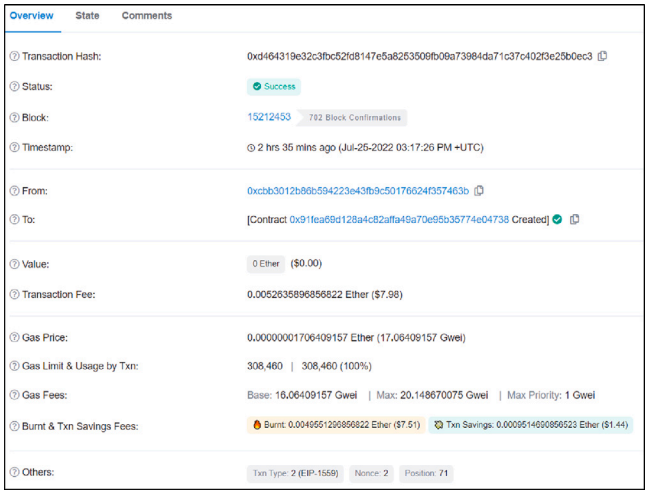


Fig. 9. The SC deployment details.

Table 7
A comparison of blockchains' transaction fees and costs ($) [46–48].

Blockchain	Tx's time	Tx's cost
Aleph Zero	~0.6 s	~0.0003 $
Avalanche	~1–2 s	1 $ for transaction fee ≤20 $ 5% for transaction fee >20 $
Digibyte	~2–3 min	~0.0005 $
Dash	~6 min	~0.2–0.3 $
Litecoin	~30 min	~0.007 $
Tezos	~30 min	~0.01 $

i7-4790 CPU, 16 GB RAM, and Ubuntu 20.04-desktop-amd64. From Table 8, T_{mul}^{ECC} and T_{add}^{ECC} are the scalar multiplication and addition operations in the ECC-based group $\mathbb{G}$. While T_{mul}^{BP}, T_{add}^{BP}, and T_{BP} are the scalar multiplication, addition, and bilinear pairing operations in BP-based group $\mathbb{G}_1$. Finally, T_{exp}, T_h, T_{enc}^{AES}, and T_{dec}^{AES} are the exponentiation, SHA-256 hashing, encryption, and decryption operations using the advanced encryption standard (AES) algorithm.

Table 9 presents a comparison of computation and communication costs between the BCSKE scheme and those described in [19–22,35].

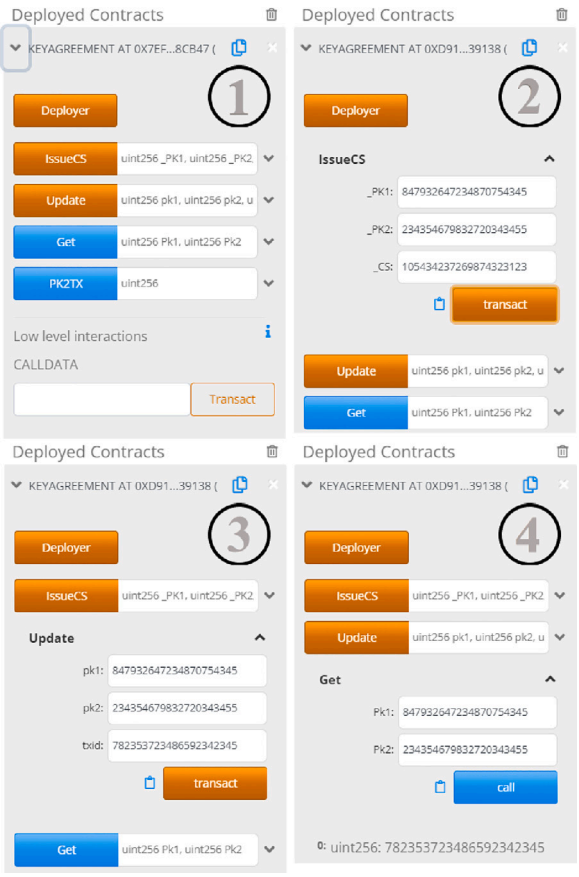


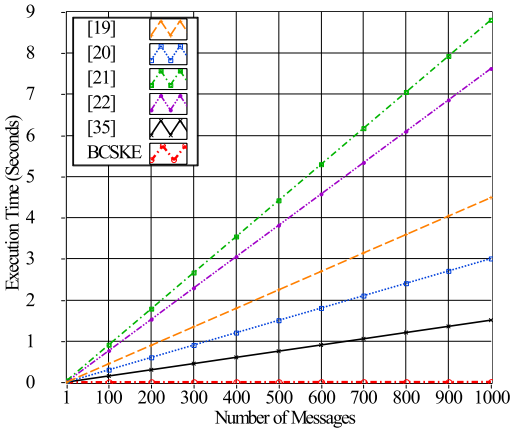
Fig. 10. An example of the *SC* functionality.

Table 8
The cost of computing different cryptographic operations in ms.

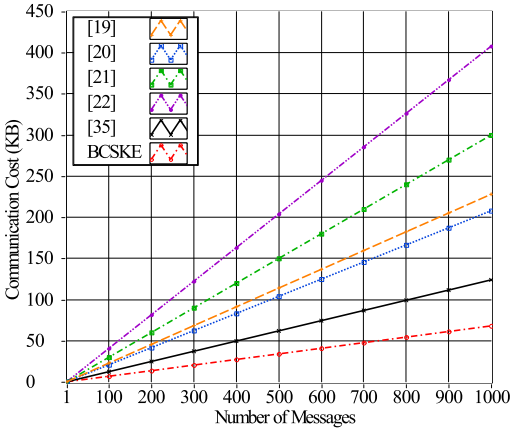
Definition	Symbol	Time
ECC-based scalar multiplication in $\mathbb{G}$	T_{mul}^{ECC}	1.489
ECC-based Point addition in $\mathbb{G}$	T_{add}^{ECC}	0.008
BP-based scalar multiplication in $\mathbb{G}_1$	T_{mul}^{BP}	2.521
BP-based point addition in $\mathbb{G}_1$	T_{add}^{BP}	0.018
BP operation in $\mathbb{G}_1$	T_{BP}	13.44
Exponentiation operation	T_{exp}	1.864
Hash function operation using SHA-256	T_h	0.003
The AES encryption operation	T_{enc}^{AES}	0.002
The AES decryption operation	T_{dec}^{AES}	0.001

As can be seen, [19] demonstrates that the vehicle needs two ECC-based scalar multiplication and two hashing operations to sign a single message. Thus, the total run time for the signature generation process is $2T_{mul}^{ECC} + 2T_h \approx 2.984$ ms. While the time cost to verify a number of n received signatures comprises $3n$ ECC-based scalar multiplication, $2n$ ECC-based addition, and $2n$ hashing operations, so the total run time for the signature verification process is $(3n)T_{mul}^{ECC} + (2n)T_{add}^{ECC} + (2n)T_h \approx 4.489n$ ms. The same goes for [20–22,35].

Our calculations for the BCSKE are based on the time consumed to send n safety-related messages, ignoring the key agreement process since it only occurs once per session. The time consumed to sign a single message includes one hashing operation and one AES-based encryption operation, so the total run time for the signature generation is $1T_{enc}^{AES} + 1T_h \approx 0.005$ ms. While the time consumed to verify n messages includes n hashing operations and n AES-based decryption operations, so the total run time for the signature verification is $(n)T_{dec}^{AES} + (n)T_h \approx 0.004n$ ms. Therefore, the computation costs for generating and verifying signatures in the BCSKE scheme are lower than those in [19]. Similarly, the computation costs in the BCSKE scheme (i.e., signature generation and verification) are lower than those of [20–22,35], as shown in Fig. 11(a).



(a) The computation cost for different numbers of messages.



(b) The communication cost for different numbers of messages.

Fig. 11. Comparison of computation and communication costs.

6.3. Comparative analysis of communication cost

This subsection evaluates the communication cost of the proposed scheme. For comparison, we use the curve equation $E : y^2 = x^3 + x \ mod \ p$ for the bilinear pairing map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ represent additive group and multiplicative group with order q_1, respectively. For the 80-bit security level, we adopted the curve type "$SS512$" [51], where p is a large prime number of order 512 bits (64 bytes) [30]. Thus, the size of the element in the BP-based group $\mathbb{G}_1$ is $2 \times 512 = 1024$ bits (128 bytes). While the size of the element in the ECC-based group $\mathbb{G}$ described in Table 3 is 320 bits (40 bytes). As for the element size in Z_q^* and $Z_{q_1}^*$, the timestamp, and the hash function output, they are respectively 160 bits (20 bytes; the security level of the 1024-bit RSA key length [52]), 32 bits (4 bytes), and 160 bits (20 bytes). The lengths of the ECC and BP parameters are summarized in Table 10 [49]. In this calculation, we only consider the size of the signatures attached to the safety-related messages since all the schemes have the same message size. In [19], the transmitted signature is the set of elements $\{\{f_i, g_i\}, B_i, K_i, R_i, \{pid_{1i}, pid_{2i}, T_i\}, T_1\}$ in which $\{pid_{1i}, K_i, B_i, R_i\} \in \mathbb{G}$, $\{f_i, g_i, pid_{2i}\} \in Z_q^*$, and T_1 and T_i are the timestamp and the expiry time of the pseudo-identity, respectively. Hence, the signature size is $4 \times 40 + 4 \times 20 + 2 \times 4 =$

Table 9
Comparative analyses of computation and communication costs.

Schemes	Computation cost in ms		Communication cost in bytes
	Signature generation	Signature verification of n messages	
Sutrala et al. [19]	$2T_{mul}^{ECC} + 2T_h$	$(3n)T_{mul}^{ECC} + (2n)T_{add}^{ECC} + (2n)T_h$	$228n$
Ming et al. [20]	$3T_{mul}^{ECC} + 2T_h$	$(2n+2)T_{mul}^{ECC} + (2n+1)T_{add}^{ECC} + (3n)T_h$	$208n$
Tan et al. [21]	$6T_{mul}^{BP} + 2T_{add}^{BP} + 3T_h$	$2T_{BP} + (2n+2)T_{mul}^{BP} + (2n)T_{exp} + (2n+3)T_h$	$300n$
Li et al. [22]	$3T_{mul}^{BP} + 2T_{add}^{BP} + 1T_h$	$(3n+2)T_{mul}^{BP} + (3n)T_{add}^{BP} + (n)T_h$	$408n$
Ogundoyin et al. [35]	$T_{mul}^{ECC} + T_{add}^{ECC} + 2T_h$	$(n+1)T_{mul}^{ECC} + (2n)T_{add}^{ECC} + (n)T_h$	$124n$
BCSKE	$1T_{\text{enc}}^{AES} + 1T_h$	$(n)T_{dec}^{AES} + (n)T_h$	$68n$

Table 10
Parameters of ECC and bilinear pairing [49].

Scheme	Curve type	Pairing	Cyclic group	Length of p	Length of a group point
ECC	$E: y^2 = x^3 + ax + b \ mod \ p$	Pairing free	$\mathbb{G}(p)$	160 bits	$\lvert\mathbb{G}\rvert$ = 320 bits
BP	$E: y^2 = x^3 + x \ mod \ p$	$\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$	$\mathbb{G}_1(p)$	521 bits	$\lvert\mathbb{G}_1\rvert$ = 1024 bits

228 bytes. In [20], the transmitted signature is the set of elements $\{\{PID_{i,1}, PID_{i,2}, T_i\}, t_i, P_i, D_i, R_i, \sigma_i\}$ in which $\{PID_{i,1}, P_i, D_i, R_i\} \in \mathbb{G}$, $\{PID_{i,2}, \sigma_i\} \in Z_q^*$, and t_i and T_i are the timestamp and the expiry time of the pseudonym, respectively. Thus, the signature size is $4 \times 40 + 2 \times 20 + 2 \times 4 = 208$ bytes. In [21], the transmitted signature is the set of elements $\{ID_i, TS_2, R_i, A_i, \mathfrak{L}_i\}$ in which $\{R_i, \mathfrak{L}_i\} \in \mathbb{G}_1$, $\{ID_i, A_i\} \in Z_{q_1}^*$, and TS_2 is the timestamp. Thus, the signature size is $2 \times 128 + 2 \times 20 + 4 = 300$ bytes. In [22], the transmitted signature is the set of elements $\{R_{ui}, K_{ui}', KG_{ui}', t_i, \rho_{ui}\}$ in which $\{K_{ui}', KG_{ui}', R_{ui}\} \in \mathbb{G}_1$, $\rho_{ui} \in Z_{q_1}^*$, and t_i is the timestamp. Hence, the signature size is $3 \times 128 + 20 + 4 = 408$ bytes. In [35], the transmitted signature is the set of elements $\{PID_i^k, R_i, \vartheta_i\ PK_i, t_i^{cur}\}$ in which $\{R_i, PK_i\} \in \mathbb{G}$, $\{PID_i^k, \vartheta_i\} \in Z_q^*$, and t_i^{cur} is the timestamp. Hence, the signature size is $2 \times 40 + 2 \times 20 + 4 = 124$ bytes. For the BCSKE scheme, the transmitted signature is the set of elements $\{T_6, TxID, \sigma_6\}$ in which the size of $TxID$ and σ_6 are each equal to 32 bytes. Therefore, the signature size is 2×32+4 = 68 bytes. Based on this analysis, we conclude that the communication cost in the BCSKE scheme is lower than that of [19–22,35], as shown in Fig. 11(b).

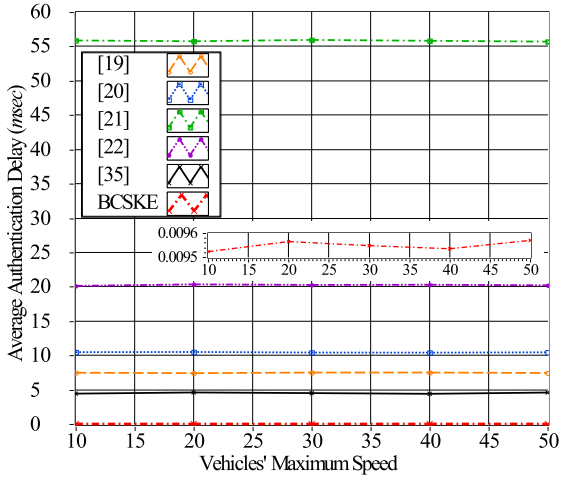
6.4. Simulation analysis

In this subsection, we use the discrete event simulator OMNeT++ 5.6.2 [53] together with a network simulator (i.e., Sumo 1.8.0 [54], INET 4.2.5 [55], and Veins 5.2 [56]) to carry out the vehicular simulation to analyze the network performance of the BCSKE scheme. The simulation uses the IEEE 802.11p standard in a 2500 × 2500 m^2 area, and all the other simulation parameters are listed in Table 11. We investigate the performance of the BCSKE scheme from the standpoint of average packet delay and packet loss ratio and compare it with those of [19–22,35].

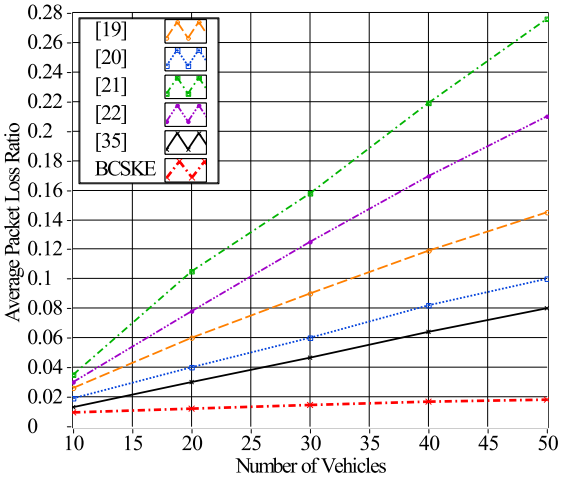
1. *Average authentication delay* (AAD): It is also called the end-to-end packet delay, which consists of the sum of the message transmission and verification delays, which can be computed using the following formula.

$$AAD = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{n_i} \sum_{j=1}^{n_i} (T_{rec}^j - T_{send}^j) \right) \tag{16}$$

where n_i is the number of received messages by the vehicle V_i, N is the total number of vehicles inside the network, and T_{send}^j and T_{rec}^j are the sending and receiving time of the message m_j. The simulation scenario is established such that a fixed number of moving vehicles (i.e., 20 vehicles) periodically send emergency traffic messages every 100 ms. In Fig. 12(a), the authentication delay for moving vehicles is shown at different speeds (i.e., 10, 20, 30, 40, 50 m/s). As can be seen from the figure, the authentication delay is approximately constant at different speeds. The



(a) Average authentication delay at different speeds.



(b) Average packet loss ratio at different densities.

Fig. 12. OMNeT++ simulation results.

varying vehicles' speeds, accelerations, and decelerations cause a slight fluctuation in the distance between the communicating vehicles, resulting in subtle variations in the average authentication delay caused by the negligible propagation delay. We find

Table 11
OMNeT++ simulation parameters.

Parameter	Value
Simulation area	2500 × 2500 m^2
Duration of simulation	200 s
Number of vehicles	10, 20, 30, 40, and 50
Vehicles' maximum speed (m/s)	10, 20, 30, 40, and 50
MAC layer protocol	802.11 p
Transmission power	50 mW
Data transmission rate	6 Mbps
Broadcasting rate	100 ms
Receiver sensitivity	−110 dBm
Noise floor, e.g., thermal noise, etc.	−98 dBm
Maximum interference distance	2500 m
Antenna type	Monopole antenna
Used channel	Control channel (CCH)
Vehicle's length	2.5 m
Minimum gap between vehicles	2 m
Vehicles' acceleration	2 m/s^2
Vehicles' deceleration	3 m/s^2

that the BCSKE scheme has the lowest average authentication delay (∼0.0095 ms) among its rivals.

2. *Average packet loss ratio* (APLR): The ratio between lost packets and transmitted packets reflects the packet loss ratio, which can be computed using the following formula.

$$APLR = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{N_{lost}^{i}}{N_{rec}^{i}+N_{lost}^{i}}\right) \quad (17)$$

where N_{rec}^{i} and N_{lost}^{i} are the number of successfully received and lost data packets from the vehicle V_i, respectively. In Fig. 12(b), the packet loss ratio for a different number of vehicles (i.e., 10, 20, 30, 40, 50 vehicles) is shown at maximum speed of moving vehicles of 20 m/s . The figure clearly shows that the packet loss ratio of all the schemes (i.e., the number of dropped packets) increases as the number of vehicles increases. However, it is worth noting that the slopes of [21,22] are much higher than that of [19,20,35] because [21,22] have bilinear pairing-based verification processes and the largest sizes of transmitted messages. While the slopes of [19,20,35] are higher than that of the BCSKE scheme because [19,20,35] have ECC-based verification processes. Based on these results, we conclude that the BCSKE scheme outperforms its competitors in [19–22,35].

7. Conclusions

In accordance with the contributions listed, this study introduces a lightweight SKC-based re-authentication method that supports forward and backward secrecy (1st contribution). For key agreement, we propose an efficient secret key extraction-based authentication scheme for VANETs, leveraging the immutability of blockchain technology to design a lightweight smart contract for reconciling the mismatched bits incurred by the channel non-reciprocity components (2nd contribution). In addition, we proved the correctness and security robustness of the BCSKE scheme using the BAN-logic and AVISPA simulator, respectively, demonstrating the scheme's resistance to common adversarial attacks (3rd contribution). We highlighted that the BCSKE scheme is effective in high-density traffic environments, and it has been demonstrated that the required time to verify 1000 messages is improved by ∼99% compared to previous studies [19–22,35]. While the communication cost is improved by 70%, 67%, 77%, 83%, and 45% in comparison to that of [19–22,35], respectively (4th contribution). Using OMNeT++, we demonstrated that the proposed scheme offers a lower authentication delay and packet loss ratio than competing approaches. Our future work will explore the possibility of incorporating group secret key extraction techniques with the consortium blockchain to develop an efficient group signature-based authentication scheme for VANETs.

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.

Data availability

Data will be made available on request.

Acknowledgment

This work was supported by the Egyptian Ministry of Defence.

Appendix

AVISPA Simulation codes are shown in this section.

Code 1: HLPSL code for the role of the vehicle V_1, played by V_1

```
role role_V1 (V1,V2,RSU,RTA:agent,KV1,KV2,KTA,KRSU,KRTA:public_key,
              KV1Rta:symmetric_key,SND,RCV:channel(dy))
played_by V1
def=
        local
                State:nat,TR,T1,T2,T4,T6,PPV1,PPV2,CS,M:text,
                K1,K2:symmetric_key
        init
                State:=0
        transition
                1. State=0 /\ RCV(start) = | > State':=1 /\ T1':=new() /\
                   SND(PPV1.T1'.KV1.TR.{KV1.TR}_inv(KTA).{PPV1.T1'
                   .KV1.TR.{KV1.TR}_inv(KTA)}_inv(KV1))
                   %% V1 hopes that PPV1 will be authenticated by V2
                   /\ witness(V1,V2,auth_1,PPV1)
                2. State=1 /\ RCV(PPV2.T2'.KV2.TR.{KV2.TR}_inv(KTA).
                   {PPV2.T2'.KV2.TR.{KV2.TR}_inv(KTA)}_inv(KV2))
                   = | > State':=2 /\ K1':=new() /\ T4':=new() /\
                   SND({K1'}_KV1Rta.KV2.TR.{KV2.TR}_inv(KTA).KV1.
                   TR.{KV1.TR}_inv(KTA).T4'.{{K1'}_KV1Rta.KV2.TR.
                   {KV2.TR}_inv(KTA).KV1.TR.{KV1.TR}_inv(KTA).T4'}
                   _inv(KV1))
                   %% V1 verifies the received PPV2 from V2
                   /\ request(V1,V2,auth_2,PPV2)
                   %% V1 believes in the secrecy of K1' transmitted to the RSU
                   /\ secret(K1',sec_1,{V1,RSU})
                   %% V1 hopes that {K1'}_KV1Rta will be authenticated
                        by the RSU
                   /\ witness(V1,RSU,auth_4,{K1'}_KV1Rta)
                3. State=2 /\ RCV(KV1.KV2.CS'.{KV1.KV2.CS'}
                   _inv(KRTA)) = | > State':=3 /\ T6':=new() /\ K2':=
                   xor(Cs',K1) /\ T6':=new() /\ SND(M.T6'.KV1.KV2.CS.
                   {KV1.KV2.CS}_inv(KRTA).{M.T6'.KV1.KV2.CS.{KV1.
                   KV2.CS}_inv(KRTA)}_K2')
                   %% V1 verifies the received CS' from the RTA
                   /\ request(V1,RTA,auth_6,CS')
                   %% V1 hopes that the message M will be authenticated
                        by V2
                   /\ witness(V1,V2,auth_7,M)
end role
```

Code 2: HLPSL code for the role of the vehicle V_2, played by V_2

```
role role_V2 (V2,V1,RSU,RTA:agent,KV1,KV2,KTA,KRSU,KRTA:public_key,
              KV2Rta:symmetric_key,SND,RCV:channel(dy))
played_by V2
def=
        local
                State:nat,TR,T1,T2,T3,T6,PPV1,PPV2,CS,M:text,
                K1,K2:symmetric_key
        init
                State:=0
        transition
                1. State=0 /\ RCV(PPV1.T1'.KV1.TR.{KV1.TR}_inv(KTA).
                   {PPV1.T1'.KV1.TR.{KV1.TR}_inv(KTA)}_inv(KV1))
                   = | > State':=1 /\ T2':=new() /\ SND(PPV2.T2'.KV2.
                   TR.{KV2.TR}_inv(KTA).{PPV2.T2'.KV2.TR.{KV2.TR}
                   _inv(KTA)}_inv(KV2)) /\ K2':=new() /\ SND({K2'}
                   _KV2Rta.KV1.TR.{KV1.TR}_inv(KTA).KV2.TR.{KV2.TR}
                   _inv(KTA).T3'.{{K2'}_KV2Rta.KV1.TR.{KV1.TR}
                   _inv(KTA).KV2.TR.{KV2.TR}_inv(KTA).T3'}_inv(KV2))
                   %% V2 verifies the received PPV1 from V1
                   /\ request(V2,V1,auth_1,PPV1)
                   %% V2 hopes that PPV2 will be authenticated by V1
                   /\ witness(V2,V1,auth_2,PPV2)
                   %% V2 believes in the secrecy of K2' transmitted to the RSU
                   /\ secret(K2',sec_2,{V2,RSU})
                   %% V2 hopes that {K2'}_KV2Rta will be authenticated
                        by the RSU
                   /\ witness(V2,RSU,auth_3,{K2'}_KV2Rta)
                2. State=1 /\ RCV(M.T6.KV1.KV2.CS'.{KV1.KV2.CS'}
                   _inv(KRTA).{M.T6.KV1.KV2.CS'.{KV1.KV2.CS'}
                   _inv(KRTA)}_K2) = | > State':=2
                   %% V2 verifies the received message M from V1
                   /\ request(V2,V1,auth_7,M)
end role
```

Code 3: HLPSL code for the role of the RSU, played by RSU

```
role role_RSU (RSU,V1,V2,RTA:agent,KV1,KV2,KTA,KRSU,KRTA:
               public_key,SND,RCV:channel(dy))
played_by RSU
def=
        local
                State:nat,TR,T3,T4,T5:text,
                K1,K2,KV1Rta,KV2Rta:symmetric_key
        init
                State:=0
        transition
                1. State=0 /\ RCV({K2'}_KV2Rta'.KV1.TR.{KV1.TR}
                   _inv(KTA).KV2.TR.{KV2.TR}_inv(KTA).T3'.{{K2'}
                   _KV2Rta'.KV1.TR.{KV1.TR}_inv(KTA).KV2.TR.{KV2
                   .TR}_inv(KTA).T3'}_inv(KV2)) = | > State':=1
                   %% RSU verifies the received {K2'}_KV2Rta' from V2
                   /\ request(RSU,V2,auth_3,{K2'}_KV2Rta)
                2. State=1 /\ RCV({K1'}_KV1Rta'.KV2.TR.{KV2.TR}
                   _inv(KTA).KV1.TR.{KV1.TR}_inv(KTA).T4'.{{K1'}
                   _KV1Rta'.KV2.TR.{KV2.TR}_inv(KTA).KV1.TR.
                   {KV1.TR}_inv(KTA).T4'}_inv(KV1)) = | > State':=2 /\
                    T5':=new() /\ SND({K1'}_KV1Rta'.{K2}_KV2Rta.KV1.
                   KV2.T5'.KRSU.TR.{KRSU.TR}_inv(KTA).{{K1'}
                   _KV1Rta'.{K2}_KV2Rta.KV1.KV2.T5'.KRSU.TR.{KRSU.
                   TR}_inv(KTA)}_inv(KRSU))
                   %% RSU verifies the received {K1'}_KV1Rta' from V1
                   /\ request(RSU,V1,auth_4,{K1'}_KV1Rta')
                   %% RSU beleives in the secrecy of K1' transmitted
                        to the RTA
                   /\ secret(K1',sec_3,{RSU,RTA})
                   %% RSU beleives in the secrecy of K2 transmitted to the RTA
                   /\ secret(K2,sec_4,{RSU,RTA})
                   %% RSU hopes that {K1'}_KV1Rta' and {K2}_KV2Rta
                        will be authenticated by RTA
                   /\ witness(RSU,RTA,auth_5,{K1'}_KV1Rta'.{K2}_KV2Rta)
end role
```

Code 4: HLPSL code for the role of the RTA, played by RTA

```
role role_RTA (RTA,V1,V2,RSU:agent,KV1,KV2,KTA,KRSU,KRTA:public_key,
               KV1Rta,KV2Rta:symmetric_key,SND,RCV:channel(dy))
played_by RTA
def=
        local
                State:nat,TR,T5,CS:text,
                K1,K2:symmetric_key
        init
                State:=0
        transition
                1. State=0 /\ RCV({K1'}_KV1Rta'.{K2}_KV2Rta.KV1.KV2.
                   T5'.KRSU.TR.{KRSU.TR}_inv(KTA).{{K1'}_KV1Rta.
                   {K2'}_KV2Rta.KV1.KV2.T5'.KRSU.TR.{KRSU.TR}
                   _inv(KTA)}_inv(KRSU)) = | > State':=1 /\ CS':=xor(K1',
                   K2') /\ SND(KV1.KV2.CS'.{KV1.KV2.CS'}_inv(KRTA))
                   %% RTA verifies the received {K1'}_KV1Rta' and {K2'}
                        _KV2Rta from the RSU
                   /\ request(RTA,RSU,auth_5,{K1'}_KV1Rta'.{K2'}_KV2Rta)
                   %% RTA hopes that CS' will be authenticated by V1
                   /\ witness(RTA,V1,auth_6,CS')
end role
```

Code 5: HLPSL code for the roles of session and environment and protocol goals

```
role session (V1,V2,RSU,RTA:agent,KV1,KV2,KTA,KRSU,KRTA:public_key,
              KV1Rta,KV2Rta:symmetric_key,SND,RCV:channel(dy))
def=
        local
                SND1,RCV1,SND2,RCV2,SND3,RCV3,SND4,RCV4:channel(dy)
        composition
                role_V1  (V1,V2,RSU,RTA,KV1,KV2,KTA,KRSU,KRTA,
                          KV1Rta,SND1,RCV1) /\
                role_V2  (V2,V1,RSU,RTA,KV1,KV2,KTA,KRSU,KRTA,
                          KV2Rta,SND2,RCV2) /\
                role_RSU (RSU,V1,V2,,RTA,KV1,KV2,KTA,KRSU,KRTA,
                          SND3,RCV3) /\
                role_RTA (RTA,V1,V2,RSU,KV1,KV2,KTA,KRSU,KRTA,
                          KV1Rta,KV2Rta,SND4,RCV4)
end role
role environment ()
def=
        const
                kv1,kv2,kta,krsu,krta:public_key,
                k1,k2,kv1rta,kv2rta:symmetric_key,
                v1,v2,rsu,rta:agent,
                auth_1,auth_2,auth_3,auth_4,auth_5,auth_6,auth_7,sec_1,sec_2,
                sec_3,sec_4:protocol_id
                intruder_knowledge={v1,v2,rsu,rta,kv1,kv2,kta,krsu,krta}
        composition
                session(v1,v2,rsu,rta,kv1,kv2,kta,krsu,krta,kv1rta,kv2rta)
end role
goal
                secrecy_of sec_1,sec_2,sec_3,sec_4
                authentication_on auth_1,auth_2,auth_3,auth_4,auth_5,auth_6,
                auth_7
end goal
environment()
```

References

[1] Manvi SS, Tangade S. A survey on authentication schemes in VANETs for secured communication. Veh Commun 2017;9:19–30.

[2] Al-Shareeda MA, Anbar M, Hasbullah IH, Manickam S. Survey of authentication and privacy schemes in vehicular ad hoc networks. IEEE Sens J 2021;21(2):2422–33.

[3] Gräfling S, Mähönen P, Riihijärvi J. Performance evaluation of IEEE 1609 WAVE and IEEE 802.11p for vehicular communications. In: 2010 second international conference on ubiquitous and future networks. ICUFN, 2010, p. 344–8.

[4] Kenney JB. Dedicated short-range communications (DSRC) standards in the united states. Proc IEEE 2011;99(7):1162–82.

[5] El-Rewini Z, Sadatsharan K, Selvaraj DF, Plathottam SJ, Ranganathan P. Cybersecurity challenges in vehicular communications. Veh Commun 2020;23.

[6] Shawky MA, Abbasi QH, Imran MA, Ansari S, Taha A. Cross-layer authentication based on physical-layer signatures for secure vehicular communication. In: 2022 IEEE intelligent vehicles symposium. IV, 2022, p. 1315–20.

[7] Raya M, Papadimitratos P, Hubaux J-P. Securing vehicular communications. IEEE Wirel Commun 2006;13(5):8–15.

[8] Shamir A. Identity-based cryptosystems and signature schemes. In: Proceedings of workshop on the theory and application of cryptographic techniques, Vol. 196. 1984, p. 47–53.

[9] Ali I, Hassan A, Li F. Authentication and privacy schemes for vehicular ad hoc networks (VANETs): A survey. Veh Commun 2019;16:45–61.

[10] Zeng K. Physical layer key generation in wireless networks: Challenges and opportunities. IEEE Commun Mag 2015;53(6):33–9.

[11] Bottarelli M, Karadimas P, Epiphaniou G, Ismail DKB, Maple C. Adaptive and optimum secret key establishment for secure vehicular communications. IEEE Trans Veh Technol 2021;70(3):2310–21.

[12] Chen S, Pang Z, Wen H, Yu K, Zhang T, Lu Y. Automated labeling and learning for physical layer authentication against clone node and sybil attacks in industrial wireless edge networks. IEEE Trans Ind Inf 2021;17(3):2041–51.

[13] Wang Q, Su H, Ren K, Kim K. Fast and scalable secret key generation exploiting channel phase randomness in wireless networks. In: 2011 proceedings IEEE INFOCOM. 2011, p. 1422–30.

[14] Bottarelli M, Epiphaniou G, Ismail DKB, Karadimas P, Al-Khateeb H. Physical characteristics of wireless communication channels for secret key establishment: A survey of the research. Comput Secur 2018;78:454–76.

[15] Baksi S, Snoap J, Popescu DC. Secret key generation using one-bit quantized channel state information. In: IEEE wireless communications and networking conference. WCNC, 2017, p. 1–6.

[16] Epiphaniou G, Karadimas P, Kbaier Ben Ismail D, Al-Khateeb H, Dehghantanha A, Choo KR. Nonreciprocity compensation combined with turbo codes for secret key generation in vehicular ad hoc social IoT networks. IEEE Internet Things J 2018;5(4):2496–505.

[17] He D, Zeadally S, Xu B, Huang X. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks. IEEE Trans Inf Forensics Secur 2015;10(12):2681–91.

[18] Li J, Choo KR, Zhang W, Kumarid S, Rodrigues J, Khan MK, Hogrefe D, Efficient EPA-CPPA:An. Provably-secure and anonymous conditional privacy-preserving authentication scheme for vehicular ad hoc networks. Veh Commun 2018;240.

[19] Sutrala AK, Bagga P, Das AK, Kumar N, Rodrigues JJPC, Lorenz P. On the design of conditional privacy preserving batch verification-based authentication scheme for internet of vehicles deployment. IEEE Trans Veh Technol 2020;69(5):5535–48.

[20] Ming Y, Cheng H. Efficient certificateless conditional privacy-preserving authentication scheme in VANETs. Mob Inf Syst (Hindawi) 2019;2019.

[21] Tan H, Chung I. Secure authentication and key management with blockchain in VANETs. IEEE Access 2020;8:2482–98.

[22] Li J, Ji Y, Choo K-KR, Hogrefe D. CL-CPPA: Certificate-less conditional privacy-preserving authentication protocol for the internet of vehicles. IEEE Internet Things J 2019;6(6):10332–43.

[23] Liu Y, Wang L, Chen H-H. Message authentication using proxy vehicles in vehicular ad hoc networks. IEEE Trans Veh Technol 2015;64(8):3697–710.

[24] Asaar MR, Salmasizadeh M, Susilo W, ajidi A. A secure and efficient authentication technique for vehicular ad-hoc networks. IEEE Trans Veh Technol 2018;67(6).

[25] Vijayakumar P, Azees M, Kozlov SA, Rodrigues JJPC. An anonymous batch authentication and key exchange protocols for 6G enabled VANETs. IEEE Trans Intell Transp Syst 2022;23(2):1630–8.

[26] Chen Y, Martinez J-F, Castillejo P, Lopez L. A bilinear map pairing based authentication scheme for smart grid communications: Pauth. IEEE Access 2019;7:22633–43.

[27] Shao J, Lin X, Lu R, Zuo C. A threshold anonymous authentication protocol for VANETs. IEEE Trans Veh Technol 2016;65(3):1711–20.

[28] Azees M, Vijayakumar P, Deboarh LJ. EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks. IEEE Trans Intell Transp Syst 2017;18:2467–76.

[29] Lim K, Liu W, Wang X, Joung J. SSKM: Scalable and secure key management scheme for group signature based authentication and CRL in VANET. Electronics 2019;8.

[30] Jiang Y, Ge S, Shen X. AAAS: An anonymous authentication scheme based on group signature in VANETs. IEEE Access 2020;8.

[31] Vijayakumar P, Azees M, Kannan A, Deborah LJ. Dual authentication and key management techniques for secure data transmission in vehicular ad hoc networks. IEEE Trans Intell Transp Syst 2016;17(4):1015–28.

[32] Xiong H, Chen J, Mei Q, Zhao Y. Conditional privacy-preserving authentication protocol with dynamic membership updating for VANETs. IEEE Trans Dependable Secure Comput 2022;19(3):2089–104.
[33] Lu Z, Liu W, Wang Q, Qu G, Liu Z. A privacy-preserving trust model based on blockchain for VANETs. IEEE Access 2018;6:45655–64.
[34] Zheng D, Jing C, Guo R, Gao S, Wang L. A traceable blockchain-based access authentication system with privacy preservation in VANETs. IEEE Access 2019;7:117716-117726.
[35] Ogundoyin SO, Kamil IA. An efficient authentication scheme with strong privacy preservation for fog-assisted vehicular ad hoc networks based on blockchain and neuro-fuzzy. Veh Commun 2021;31.
[36] Lu Z, Wang Q, Qu G, Zhang H, Liu Z. A blockchain-based privacy-preserving authentication scheme for VANETs. IEEE Trans Very Large Scale Integr (VLSI) Syst 2019;27(12):2792–801.
[37] Ma Z, Zhang J, Guo Y, Liu Y, Liu X, He W. An efficient decentralized key management mechanism for VANET with blockchain. IEEE Trans Veh Technol 2020;69(6):5836–49.
[38] Azees M, Vijayakumar P, Deborah LJ, Marimuthu K, Subaja CM. BBAAS: Blockchain-based anonymous authentication scheme for providing secure communication in VANETs. Secur Commun Netw (Hindawi) 2021;2021.
[39] Lin C, He D, Huang X, Kumar N, Choo KR. BCPPA: A blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks. IEEE Trans Intell Transp Syst 2021;22(12).
[40] Son S, Lee J, Park Y, Park Y, Das AK. Design of blockchain-based lightweight V2I handover authentication protocol for VANET. IEEE Trans Netw Sci Eng 2022;9(3):1346–58.
[41] Sharma TK. Types of blockchains explained- public vs. private vs. consortium. Blockchain Council; 2022.
[42] Ghosh BC, Bhartia T, Addya SK, Chakraborty S. Leveraging public-private blockchain interoperability for closed consortium interfacing. In: IEEE INFOCOM 2021 - IEEE conference on computer communications. 2021, p. 1–10.
[43] Research Certicom. Standards for efficient cryptography. In: SEC 2: Recommended elliptic curve domain parameters 1.0. 2000, p. 9–10.
[44] Burrows M, Abadi M, Needham R. A logic of authentication. ACM Trans Comput Syst 1990;8(1):18–36.
[45] Armando A, Basin D, Boichut Y, Chevalier Y, Compagna L, Cuéllar J, Mödersheim S. The AVISPA tool for the automated validation of internet security protocols and applications. In: International conference on computer aided verification. Springer; 2005, p. 5–281.
[46] Raczyński M. What is the fastest blockchain and why? Analysis of 43 blockchains. Aleph Zero Website; 2021, available at: https://alephzero.org/blog/what-is-the-fastest-blockchain-and-why-analysis-of-43-blockchains.
[47] Butler C. Top 10 blockchains with transaction fees under 1$. DappRadar Website; 2021, available at: https://dappradar.com/blog/top-10-blockchains-with-transaction-fees-under-1.
[48] Ahmad K. 10 cryptocurrencies with almost zero transaction fees. MUO Website; 2021, available at: www.makeuseof.com/cryptocurrencies-with-almost-zero-transaction-fees.
[49] Al-shareeda MA, Anbar M, Manickam S, Hasbullah IH. An efficient identity-based conditional privacy-preserving authentication scheme for secure communication in a vehicular ad hoc network. Symmetry 2020;12(10).
[50] Miracl library. 2022, Available at: https://github.com/miracl/MIRACL Accessed: (2022, June).
[51] Cui H, Wan Z, Deng RH, Wang G, Li Y. Efficient and expressive keyword search over encrypted data in cloud. IEEE Trans Dependable Secure Comput 2018;15(3):409–22.
[52] Barker E. Recommendation for key management. NIST special publication 800-57 part 1 revision 5, 2020, p. 53–4.
[53] Omnet++ discrete event simulator. 2019, Available at: https://omnetpp.org/ Accessed: (2019, November).
[54] Simulation of urban mobility. 2022, Available at: https://sumo.dlr.de/docs/index.html Accessed: (2022, June).
[55] INET framework. 2022, Available at: https://inet.omnetpp.org/2021-05-18-INET-4.3.2-released.html, Accessed: (2022, June).
[56] The open source vehicular network simulation framework. 2022, Available at: https://veins.car2x.org Accessed: (2022, June).

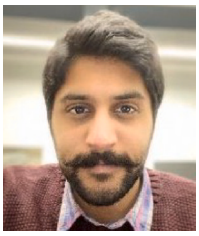
Mahmoud A. Shawky was born in 1990 in Saudi-Arabia. He received his B.Sc. degree in Electronics and Electrical Engineering in 2012 from Air Defence College, Alexandria University, M.Sc. (Eng.) degree in Authentication Mechanisms in Computer Network Protocols from Alexandria University, Alexandria, Egypt. He is currently a lecturer in the Egyptian Air Defence College. He is currently pursuing a Ph.D. degree in James Watt School of Engineering, University of Glasgow, UK. His research interests are in the area of cryptography and number theory, digital signatures, authentication in wireless communications and cyber security.

Dr. Muhammad Usman (Senior Member, IEEE) is a Lecturer in the Department of Electrical and Electronic Engineering at Glasgow Caledonian University (GCU), UK. Before Joining GCU, he was a Research Associate at the University of Glasgow, UK, in the EPSRC funded COG-MHEAR program grant. This transformative research aims to re-design current hearing aids and assistive technology, where his role was to integrate end-user context in visually-assisted hearing-aid design in a privacy-preserving manner. He is endorsed by the Royal Academy of Engineering as Global Talent. He holds a Ph.D. degree (cum laude) in information and communication technologies from the University of Trento, Italy. He has as exceptional research output record in the field of 5G and beyond 5G cellular systems, radio frequency (RF) sensing, network security, Internet-of-Things (IoT) and developing intelligent systems for healthcare sector. He is senior member of IEEE. His research interests include RF sensing, wireless communication, software-defined networks, cyber security and assistive technologies for intelligent healthcare systems.

Dr. David Flynn is a Professor of Cyber Physical Systems at the University of Glasgow. David is the founder of the Smart Systems Group (SSG) and Associate Director of the UKs National center for Energy Systems Integration. He has expertise in Systems Engineering and Cyber Physical Systems (CPS). He is a Visiting Professor at Heriot-Watt University and also the Vice-Chair of the Institute of Engineering and Technology (IET) Scotland. Professor Flynn's research explores the challenges, opportunities, and underpinning technologies within Cyber Physical Systems as to promote sustainability and create resilience in systems, organizations, networks and societies. His-degrees include a BEng (Hons), 1st Class in Electrical and Electronic Engineering (2002), an MSc (Distinction) in Microsystems (2003) and a Ph.D. in Microscale Magnetic Components (2007), from Heriot-Watt University, Edinburgh. David teaches Smart System Integration, Electrical Engineering and Energy Systems.

Dr. Muhammad Ali Imran is Professor of Communication Systems at the University of Glasgow, Dean of University of Glasgow UESTC, and Head of the Communications, Sensing and Imaging group (110^{+} researchers). He is an Affiliate Professor at the University of Oklahoma, USA, a visiting Professor at the 5G Innovation Centre of Institute for Communication Systems at the University of Surrey, UK and an Affiliate Research Professor with AI Research Centre of Ajman University, UAE. He has led a number of multimillion-funded international research projects (grant income £15M over last 10 years) encompassing the areas of energy efficiency, fundamental performance limits, sensor networks and self-organizing cellular networks. He also led the new physical layer work area for 5G innovation centre at Surrey University. He has a global collaborative research network spanning both academia and key industrial players in the field of wireless communications. He has taught on international short courses in USA, Pakistan, Tunisia, Malaysia and China. He has published over 400 peer-reviewed research papers. He has been awarded the IEEE Comsoc's Fred Ellersick award 2014 and FEPS Learning and Teaching award 2014. He has given an invited TEDx talk (2015) and more than 20 plenary talks and panels at international conferences. He was cochair of the NGNI Symposium of IEEE ICC 2019, and the founder of IEEE Workshop BackNets 2015. He has chaired several tracks/workshops at international conferences including IWCMC, Global SIP, Crowncom, European Wireless, Stemcom 5G, ICC, VTC. He has been a guest editor for IET Communications, IET Signal Processing, IEEE Communications Magazine, IEEE Wireless Communication Magazine, IEEE Access and IEEE JSAC. He is Associate Editor for IEEE Transactions on Communications and IEEE Open Access. He is a senior member of the IEEE and a Senior Fellow Higher Education Academy (SFHEA), UK. He received his Ph.D. Degree from Imperial College London, UK.

Dr. Qammer H. Abbasi is a Reader with the James Watt School of Engineering, University of Glasgow, Deputy Head for the Communication Sensing and Imaging group (110^+ researchers), Deputy Theme Lead for Quantum & Nanotechnology in the University's Advance Research Centre, Co-Manager of the RF and Terahertz Laboratory, Lead for Healthcare and Internet of things use cases with the Scotland 5G Center Urban Testbed Program and Director for the Dual Ph.D. Degree program, UK. He has a grant portfolio of £6M and has contributed to more than 350^+ leading international technical journal and peer reviewed conference papers, as well as ten books. He is Associate Editor for IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology, IEEE Sensors, IEEE Internet of Things and IEEE open access Antenna and Propagation, Senior Editor for Frontiers IoT and Sensors Networks section. He is a committee member for IEEE APS Young professional, Sub-committee Chair for IEEE YP Ambassador program, IEEE 1906.1.1 standard on nano communication, IEEE APS/SC WG P145, IET Antenna & Propagation and healthcare networks. He has been a member of the technical program committees of several IEEE flagship conferences and acted as TPC chair and executive chair for 4th, 5th and 6th international UCET conference 2019, 2020, 2021 in addition to being General Chair of EAI Bodynets 2021. He is an expert reviewer for the UK National Commission for UNESCO's, EPSRC and MRC UK (panel member, 2020), Qatar national research funds, Flemish funding council (FWO, panel member), Belgium, OSF Poland, British council, UAE and KSA funds. He serves regularly as an organizer of conferences, special sessions, workshops and TPC member for several IEEE flagship conferences. He has received several research recognitions including the URSI 2019 Young Scientist Award, the UK Exceptional Talent Endorsement by the Royal Academy of Engineering, the Sensor 2021 Young Scientist Award, the National Talent Pool Award in Pakistan, the International Young Scientist Award by NSFC China, the National Interest Waiver by USA, the University Research Excellence Award from TAMUQ for two consecutive years, the Reward for Excellence from the University of Glasgow, the Research Culture Award from the University of Glasgow, and the Pakistan Award for services to Antenna and RF community. In addition, his work has received media coverage from BBC news, Scotland TV, Analog IC tips, Microwaves & RF newsletters, Vertical news, Pakistan Dawn News, Fiercewireless, City42, Dunya news and Chinese news. He is an IEEE senior member and Chair of the IEEE AP/MTT Scotland joint chapter as well as a Fellow of the Royal Society of Arts.



Dr. Shuja Ansari (SMIEEE) is a Lecturer in the Glasgow College UESTC at the University of Glasgow, UK. He received the M.Sc. degree (distinction) in Telecommunications Engineering in 2015, and the Ph.D. degree in Engineering in 2019 from Glasgow Caledonian University (GCU), UK. He joined the Communications, Sensing and Imaging research group at the University of Glasgow as a Research Associate in 2019 working on energy efficient 5G mobile networks. He is currently the Wave-1 Urban 5G use case implementation lead at Glasgow 5G Testbed funded by the Scotland 5G Centre of the Scottish Government working on a variety of 5G applications in partnership with several industrial and academic partners. His research interests include Wireless Communications, Internet of Things, Cooperative Intelligent Transport Systems, Autonomous Systems, Terrestrial/Airborne Mobile Networks, and Healthcare technologies.

Dr. Ahmad Taha (MIEEE) is a Lecturer in the Glasgow College UESTC at the University of Glasgow, UK. He is Endorsed by the Royal Academy of Engineering as an Exceptional Promise under the Global Talent scheme, a Fellow of Advanced Higher Education (FHEA), and a UKCGE recognized Associate Supervisor. He first joined the University of Glasgow as a Postdoctoral Research Associate working on the 5G New Thinking project (funded by the Department for Digital, Culture, Media and Sport DCMS), in collaboration with several industrial and academic partners including Cisco and the University of Strathclyde, He was nominated for the Energy Institute award in 2019 due to contributions in technology-based energy-saving systems in the NHS. His research interests include smart energy systems, energy conscious networks and healthcare technology. He holds a Ph.D. Degree on (topic) from (institute) which was partially funded and in collaboration with the Medway NHS Foundation Trust in Kent, UK.