# AAKE-BIVT: Anonymous Authenticated Key Exchange Scheme for Blockchain-Enabled Internet of Vehicles in Smart Transportation

Akhtar Badshah, Muhammad Waqas, *Senior Member, IEEE*, Fazal Muhammad, Ghulam Abbas, *Senior Member, IEEE*, Ziaul Haq Abbas, Shehzad Ashraf Chaudhry, *Senior Member, IEEE*, and Sheng Chen, *Fellow, IEEE*

***Abstract*— The next-generation Internet of vehicles (IoVs) seamlessly connects humans, vehicles, roadside units (RSUs), and service platforms, to improve road safety, enhance transit efficiency, and deliver comfort while conserving the environment. Currently, numerous entities communicate in the IoVs environment via insecure public channels that are susceptible to a variety of security assaults and threats. To address these security challenges, we design an anonymous authenticated key exchange mechanism for the IoVs in smart transportation supported by blockchain, referred to as AAKE-BIVT. AAKE-BIVT securely transmits traffic information to a cluster head, before heading to a nearby RSU utilizing the established secret session keys via mutual authentication and key agreement. A cloud server (CS) then securely aggregates data from related RSUs and generates transactions. The CS combines the transactions into blocks in a peer-to-peer network of CSs, and the blocks are confirmed and added to the blockchain via a voting-based consensus method. By means of rigorous informal security studies and formal security analysis through the random oracle model, we reveal that the proposed AAKE-BIVT is resistant to a broad range of potential security assaults in the IoVs environment. Furthermore, a comparative study reveals that AAKE-BIVT outperforms existing state-of-the-art techniques, in terms of security and functionality while being more efficient in terms of communication and computation. Additionally, the blockchain simulation validates the implementation viability of our proposed AAKE-BIVT.**

***Index Terms*— Internet of Vehicles, blockchain, security, authentication, key exchange, PUF.**

Manuscript received 11 January 2022; revised 18 July 2022 and 10 October 2022; accepted 1 November 2022. Date of publication 14 November 2022; date of current version 8 February 2023. This work was supported in part by Abu Dhabi University, Abu Dhabi, United Arab Emirates. The Associate Editor for this article was P. Wang. *(Corresponding author: Shehzad Ashraf Chaudhry.)*

Akhtar Badshah is with the Telecommunications and Networking (TeleCoN) Research Center, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan, and also with the Department of Software Engineering, University of Malakand, Chakdara, Dir Lower 18800, Pakistan (e-mail: akhtarbadshah@giki.edu.pk).

Muhammad Waqas is with the Computer Engineering Department, College of Information Technology, University of Bahrain, Sakhir 32038, Bahrain, and also with the School of Engineering, Edith Cowan University, Perth, WA 6027, Australia (e-mail: engr.waqas2079@gmail.com).

Fazal Muhammad is with the Department of Electrical Engineering, University of Engineering and Technology, Mardan 23200, Pakistan (e-mail: fazal.muhammad@uetmardan.edu.pk).

Ghulam Abbas and Ziaul Haq Abbas are with the TeleCoN Research Center, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan (e-mail: abbasg@giki.edu.pk; ziaul.h.abbas@giki.edu.pk).

Shehzad Ashraf Chaudhry is with the Department of Computer Science and Information Technology, College of Engineering, Abu Dhabi University, Abu Dhabi, United Arab Emirates (e-mail: ashraf.shehzad.ch@gmail.com).

Sheng Chen is with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: sqc@ecs.soton.ac.uk).

Digital Object Identifier 10.1109/TITS.2022.3220624

## I. Introduction

SINCE its inception a few years ago, the Internet of vehicles (IoVs) has emerged as an enabling component for intelligent transportation systems (ITS). IoVs rely on a new generation of information and communication technologies to connect cars, and are heavily dependent on the Internet of Things (IoT) [1], [2], [3] to function.

In IoVs, pedestrians, cars, roadside units (RSUs), and service platforms are all considered nodes in an integrated information network, relying on wireless communication to coordinate their interactions with each other and the environment. This information network enhances the overall intelligence of the vehicles. It provides users with an efficient, safe, and convenient driving experience and traffic services while simultaneously enhancing the performance of traffic operations and increasing the insightful level of intelligent traffic services provided by the vehicles [4], [5]. It is predicted that the IoVs market value will expand by 215 percent by 2024 due to rising road safety standards and security aspects of intelligent vehicles, according to Allied Market Research [6].

With the rapid expansion of vehicular services and applications, it is anticipated that an increasing number of intelligent vehicles will produce and exchange vast quantities of data, resulting in enormous network traffic that must be managed. Additionally, the IoV's heterogeneity, high mobility, context complexity, and low latency will pose significant challenges when directly employing conventional cloud-based storage and management. Moreover, ensuring robust interoperability and compatibility between IoV entities from various service providers is challenging. Therefore, the data interchange and storage infrastructure for IoVs must be distributed, decentralized, interoperable, scalable, and flexible to accommodate future IoV growth and realize the full potential of ITS. As the platform is decentralized and distributed, it is naturally susceptible to more cyber threats; consequently, it is

crucial to protect the security, privacy, and dependability of IoV data [7], [8]. Therefore, research has been conducted into implementing blockchain as a system platform to meet the IoV's information exchange requirements. Blockchain-enabled IoV applications are believed to possess a variety of desirable characteristics, including security, decentralization, immutability, transparency, and automation [9].

Due to the nature of insecure communication via wireless channels among numerous connected entities in the IoVs' setting leads to various security vulnerabilities, and the transmitted data can be tampered by adversary $\mathcal{A}$ in various manners. Specifically, $\mathcal{A}$ can launch numerous potential security attacks, including physical device capture, replay, ephemeral secret leakage (ESL), impersonation, privileged insider (PI), denial-of-service (DoS), man-in-the-middle (MitM), and so on. Apart from these attacks, it is essential to preserve the untraceability and anonymity features of IoVs so that $\mathcal{A}$ cannot trace the communicating entity. Against the bulk of such assaults, a robust and effective authenticated key exchange (AKE) scheme is the primary line of defense. Using this technique, vehicles, RSUs, and cloud servers (CSs) can authenticate each other and generate session keys for secure communication. Furthermore, blockchain consensus mechanisms are often combined with key agreement schemes to create shared secret session keys for guaranteeing the security of communicated sessions. This is to ensure that only authorized CSs play a part in the consensus mechanism for block verification and addition to the blockchain center while simultaneously minimizing the latency and overhead issues.

Over the last few years, several AKE schemes have been proposed for the IoVs environment. However, most of the existing schemes have multiple deficiencies. Firstly, the communication and computational overheads carried out by cryptographic operations in the existing schemes are not low. Secondly, numerous schemes are not resilient enough to protect data at rest and in transit. Thirdly, most existing schemes in the ITS communication environment do not render anonymity, untraceability, and non-linkability, which are crucial security traits.

The objective of this work is to solve the aforementioned limitations of the existing schemes. We devise an anonymous AKE for blockchain-enabled IoVs in smart transportation, called AAKE-BIVT, with three levels of AKE schemes for session key establishment, namely, a) between cluster head (CH) and nearby RSU (CH2RSU), b) between two neighboring vehicles (V2V), and c) between RSU and CS (RSU2CS). These AKE schemes enable vehicles, RSUs, and CSs to authenticate and establish a session key for secure communication. For the consensus mechanism among the CSs, pairwise secret keys are utilized. Additionally, blockchain technology is indispensable for such a communication environment because it is decentralized, tamper-proof, anonymous, and robust against numerous information security assaults. These schemes, therefore, permit IoV entities to transmit and store their data secretly.

The main contributions of this paper are as follows.

1) We devise a blockchain-enabled secure communication design for smart transportation system, called AAKE-BIVT, which simultaneously permits the AKE scheme among V2CH, CH2RSU, and RSU2CS. These AKE schemes enable vehicles, RSUs, and CSs to authenticate and establish a session key for secure communication. For the consensus mechanism among the CSs, pairwise secret keys are utilized. Moreover, blockchain technology makes our proposed AAKE-BIVT more secure, reliable, and decentralized.
2) We use ultra-lightweight cryptography technology, composed of hash function, bitwise exclusive OR (XOR) operator, elliptic curve cryptography (ECC), and symmetric encryption/decryption along with physical unclonable function (PUF) to design our proposed AAKE-BIVT so that the communication and computational overheads brought by AKE procedures are reduced. A rigorous security analysis utilizing informal security analysis and the Real-Or-Random (ROR) oracle model reveals that our proposed AAKE-BIVT is resilient against potential security attacks and satisfies session-key security. The PUF feature enables smart vehicles and RSUs to prevent tampering from physical attacks.
3) We perform an extensive comparative analysis, which demonstrates that our proposed AAKE-BIVT provides enhanced security, adds additional functionality traits, and has lower computation and communication overheads than the other benchmark schemes.
4) Moreover, AAKE-BIVT blockchain solution is implemented to evaluate the performance by varying the number of mined blocks and the number of transactions per block.

The remainder of this paper is organized as follows. Section II introduces the related work for securing the IoVs network. Section III presents the background, including the network and threat models, design objectives, and relevant preliminaries. Section IV details our proposed AAKE-BIVT. In Section V, a comprehensive security analysis of the devised AAKE-BIVT is presented. Blockchain implementation and simulation results are presented and discussed in Section VI. The performance analysis of the proposed AAKE-BIVT compared with other state-of-the-art benchmark schemes is briefly discussed in Section VII. The paper is concluded in Section VIII.

## II. Related Work

Mollah et al. [8] surveyed the blockchain solution in the smart transportation environment, considering various perspectives, mechanisms, benefits, and challenges. Moreover, they presented IoVs paradigms integrating blockchain towards establishing future ITSs. They also discussed blockchain applications in the IoV's setting, including data protection and management, forensic application, ride-sharing, data and resource trading, content broadcasting, vehicle management, and traffic control and management.

Bagga et al. [10] presented a survey work, highlighting security requirements and numerous possible potential attacks in the IoVs environment. They discussed system models, taxonomy of security schemes, comparative analysis mechanisms,

TABLE I
SUMMARIZING EXISTING AKE SCHEMES

| Reference | Operations | Limitations |
|---|---|---|
| Liu *et al.* 2018 [11] | Bilinear pairings, modular exponentiation, ECC, and hash functions | • Blockchain security solution is not considered<br>• Exposed to ESL attack<br>• Computational overhead is high<br>• Dynamic node addition is not considered |
| Tan and Chung 2020 [19] | Bilinear pairings, modular exponentiation, ECC, and hash functions | • Exposed to ESL and PI attacks<br>• Does not render anonymity<br>• High computational overhead<br>• Dynamic node addition is not considered |
| Moghadam *et al.* 2020 [20] | ECC, symmetric key encryption, and hash functions | • Exposed to ESL and PI attacks<br>• Blockchain security solution is not considered |
| Li *et al.* 2020 [21] | ECC and hash functions | • High communication overhead<br>• Blockchain security solution is not considered |
| Vasudev *et al.* 2020 [22] | Symmetric key encryption and hash functions | • Unsafe against impersonation, MitM, and secret key disclosure attacks<br>• Blockchain security solution is not considered |
| Vangala *et al.* 2021 [23] | ECC, hash functions, modular addition and multiplication | • Anonymity and untraceability features are not considered |
| Chattaraj *et al.* 2021 [24] | ECC, bivariate polynomial, ECC-based signature, and hash functions | • High communication and computation overheads<br>• Anonymity and untraceability features are not considered |

**Note:** Elliptic-curve cryptography (ECC), privileged insider (PI), man-in-the-middle (MitM), ephemeral secret leakage (ESL).

various testbeds implementations, as well as various open challenges and issues related to data security in IoVs.

Table I outlines several existing AKE schemes in terms of their cryptographic operations as well as their limitations and drawbacks. To be more specific, Liu et al. [11] designed an AKE scheme for the IoVs based on certificateless short signature, where vehicles interact with their associated RSUs. However, the scheme is exposed to ESL attacks and requires a high computational overhead. Furthermore, blockchain technology and dynamic node addition are not considered. The authors of [12] devised an efficient AKE scheme for vehicular ad hoc networks (VANETs), which offers an efficient revocation mechanism for the malicious entity in the VANET system. Furthermore, their approach is computationally efficient in terms of the certificate and signature verification process. Vijayakumar et al. [13] designed a secure scheme for IoT-based health systems, which ensures location privacy for patients and doctors. They utilize the chinese remainder theorem to protect location privacy. The authors of [14] devised a blockchain-based anonymous authentication scheme, which authenticates the legitimacy of vehicle users, and a handover authentication scheme, which reduces the overhead caused by the reauthentication of vehicles. Blockchain is utilized to assure the security of the authentication codes of vehicles and realize the traceability of malicious vehicles. However, these schemes [12], [13], [14] are computationally expensive due to the utilization of bilinear pairings operations. To avoid the high computational overhead of bilinear pairing operations, Wei et al. [15] proposed a tree-based AKE scheme for securing vehicle-to-infrastructure and V2V communications in VANETs. They employ hash functions and ECC cryptography in their proposed scheme to reduce the communication and computational overheads brought by the AKE phase. Wei et al. [16] devised a lightweight AKE scheme with multi-trusted authority for fog-based VANET. Their scheme ensures security in the VANETs environment by utilizing Lagrange interpolation theorem and hash and pseudo-random functions. Moreover, their proposed scheme supports the credential revocation mechanism to achieve conditional privacy protection. In addition, the single-point failure issue is also fixed by considering the multi-trusted authority model. The scheme designed by Vinoth et al. [17] for the industrial IoT environment is lightweight due to the utilization of XOR operation, hash function, and symmetric cryptography. However, their scheme is vulnerable to replay, DoS, and sensor node capture attacks. Xia et al. [18] developed a

cloud-assisted trustworthiness evaluation procedure and efficient anonymous AKE protocol based on non-interactive zero-knowledge to assure IoT devices' privacy protection and data security in smart cities. However, the blockchain security solution is not considered to provide superior security. Tan and Chung [19] devised a secure AKE and key management with blockchain in VANETs. Their scheme ensures security in the IoVs environment by utilizing bilinear pairings, ECC, and modular exponentiation. However, the scheme is vulnerable to ESL and PI attacks. Moreover, the scheme does not preserve the anonymity feature. Furthermore, an AKE mechanism devised by Moghadam et al. [20] is exposed to ESL and PI attacks and does not support blockchain solutions. Li et al. [21] suggested a hierarchical authentication protocol for vehicular networks. They utilize ECC and hash functions for secure communications. However, their scheme does not support blockchain technology and imposes a high communication overhead. The scheme of Vasudev et al. [22] is vulnerable to secret key disclosure, impersonation, and MitM attacks. Additionally, blockchain technology is not supported by the scheme. Vangala et al. [23] devised an AKE scheme for a blockchain-enabled IoVs environment. The scheme securely communicates accidental notifications among IoVs entities and can transmit valuable information to the blockchain network for consensus. However, the scheme does not render anonymity and untraceability features. Chattaraj et al. [24] devised a certificateless key agreement scheme for blockchain-enabled smart transportation systems. However, they do not discuss key management among the cloud servers. Additionally, their proposed scheme imposes high communication and computation overheads and does not support features such as anonymity and untraceability.

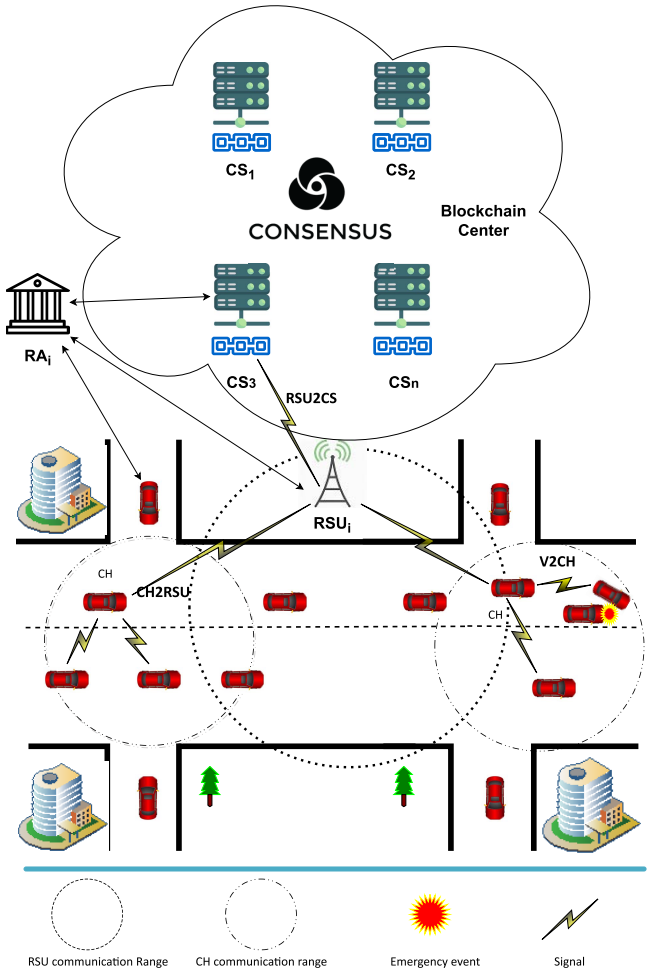


Fig. 1. Network structure for the proposed AAKE-BIVT.

## III. Background

In this section, we comprehensively discuss our network and threat models. Then, we briefly describe the design objectives and relevant preliminaries.

### A. Network Model

This subsection presents a network model of AAKE-BIVT, as depicted in Fig. 1. The network consists of several entities, such as trusted registration authorities (RAs), smart vehicles, CHs, RSUs, and CSs. In addition, it is assumed that each vehicle contains an OBU. The description of each network's entity is as follows.

- **Trusted Registration Authority:** An RA is in-charge of registering all the deployed smart vehicles, RSUs, and CSs. An RA is a completely trustworthy entity in the network. Each entity is preloaded with essential credentials following a successful registration before being deployed or put into the ITS environment.
- **Smart Vehicle:** Each vehicle contains an OBU. The OBUs have limited computing capabilities and enable vehicles to communicate with RSUs (V2RSUs) and other vehicles (V2V). The $j$th OBU, $OBU_j$ in V2RSU communication or V2V communication is responsible for transmitting and acquiring safety information, such as traffic jams, accident alerts, and shortest route identification. It can also be used for non-safety information, such as infotainment messages and toll-collection payment-related messages.
- **Cluster Head:** The vehicles in the network create numerous clusters dynamically, where a specific vehicle can be chosen as CH from a set of vehicles in the network. The CH manages and coordinates with its cluster members (CMs) when the CH has a message or data that can be forwarded to the associated RSU via a public channel, such as information about accidents, traffic, road conditions, etc. It is worth noting that for constructing distinct clusters from vehicles on the fly, a dynamic clustering approach was suggested by Kakkasageri and Manvi [25] is adopted. In their approach, vehicles traveling on the same lane segment that ends at the intersection can be considered for the cluster formation process. Each vehicle may come across its neighbors traveling at about the same speed in the same direction and on the same lane segment. The vehicles then become the best candidates to make up for any potential cluster that might emerge in that lane. A vehicle that occupies the front position on the lane is said to be an initiator since it needs to start the cluster formation process. Based on the vehicles' relative speeds and

directions of movement, the initiator vehicle chooses the vehicles. The CMs are considered to be the neighboring vehicles whose corresponding speed differential is less than the specified threshold value. To this end, every vehicle gathers the required information, including vehicle identity, speed, position, connectivity degree (number of connected vehicles), and Time-to-leave the lane segment. Time-to-leave is the length of time a vehicle remains in a particular lane segment. Consequently, an initiator selects the CH among the CMs by calculating the stability metric of each CM. The stability metric consists of average speed, connectivity degree, and Time-to-leave. The vehicle with the highest stability metric is selected as the CH.

- **Roadside Unit:** RSU is a crucial component of IoVs, which is responsible for collecting non-safety and safety-related messages from the respective CH(s). The CH obtains the information from its CMs. Next, RSU transmits the message to the corresponding CS containing the received information from the CH.
- **Blockchain Center:** In the blockchain center, the CSs form a peer-to-peer (P2P) network, named P2P cloud servers (CSN). After receiving the data from RSU, the corresponding CS treats the data as a transaction and puts it into the global transaction pool, which is available to all peer CSs. When the number of transactions hits a certain threshold, a leader is chosen from the CSs, and the block is created from the list of transactions. The block is subsequently verified and added to the blockchain by the leader using a voting consensus method based on the ripple protocol consensus algorithm (RPCA) [26].

On the CSN, the ITS environment's data is stored as a private blockchain. Utilizing blockchain technology protects against data disclosure and modification attacks. The following secure communications forms occur per the stated network model: V2CH, CH2RSU, and RSU2CS communication. The vehicles in each cluster communicate using dedicated short-range communications under the proposed AAKE-BIVT. However, such communication is vulnerable to the adversary and can be compromised due to the openness of wireless channels.

### B. Threat Model

Under the Dolev-Yao (DY) threat model [27], adversary $\mathcal{A}$ can eavesdrop, intercept, store, forge, and send messages in the network. $\mathcal{A}$ can also participate in the protocol's operation just as the most legitimate protocol participants. In general, under the DY model, $\mathcal{A}$ can control the entire network completely. Furthermore, $\mathcal{A}$ can physically capture RSU, the OBU of a vehicle, and the smart card of a vehicle owner. $\mathcal{A}$ can launch power analysis (PA) attacks [28] and try to retrieve the secret credentials kept on those seized devices.

We also utilize the current de-facto, i.e., the "Canetti and Krawczyk (CK) adversary model" [29] to devise the AAKE-BIVT security scheme. Under the CK model, apart from $\mathcal{A}$'s capability in the DY model, $\mathcal{A}$ can capture long-term private key, random number leakage, session secret key leakage, state agreement leakage through session hijacking attacks, or information stored in an insecure memory.

In the IoVs network, the trusted registration authorities and CSs are considered fully trusted and semi-trusted entities, respectively. On the other hand, the end-point communicating entity is not considered trustworthy.

### C. Design Objectives

The proposed AAKE-BIVT aims to accomplish the following primary design objectives:

- **Mutual authentication:** The communicating entities, i.e., the vehicles, RSUs, and CSs of the AKE scheme, must authenticate each other at the time of the AKE procedure to verify the legitimacy of the involved entity and the integrity of the received message.
- **Confidentiality:** The session key, created via the AKE procedure, should remain confidential for any entity except for the involved entities.
- **Untraceability:** From the viewpoint of the adversary, the AKE message transmitted from the communicating entity, i.e., vehicle, RSU, and CS, should not be traceable.
- **Non-linkability:** The scheme must guarantee non-linkability for multiple messages from the same source, i.e., there should be no correlation between different interactions of the same entity so that the adversary cannot extract sensitive credentials from different interactions of the same entity.
- **Anonymity:** The real identities of the communicating entities, i.e., vehicles, RSUs, and CSs, should be protected.
- **Resistance to potential security attacks:** The scheme must resist common attacks in the communicating environment, including MitM attacks, replay attacks, ESL attacks, impersonation attacks, data modification attacks, and physical attacks.

### D. Preliminaries

We now provide a brief overview of the preliminaries utilized in deriving the proposed AAKE-BIVT scheme.

*1) Physical Unclonable Function:* Based on the physical microstructure of a semiconductor device, the physical unclonable functions (PUFs) assign an input uniquely to an output. The challenge and response pair is an alias for the (input, output) pair for PUF. A $PUF(\cdot)$ is described by $R = PUF(C)$, where $R$ and $C$ represent response and challenge parameters, respectively [30]. A PUF circuit must reveal the characteristics listed below:

- The response generated by a PUF is dependent on the microstructure of the device.
- Response of PUF must be unique, reliable, difficult to predict, and easy to implement and test.
- The PUF circuit could not be copied/cloned.

*2) Fuzzy Extractor:* Even though PUF circuits are highly reliable, noise and temperature variation can cause deviation in PUF output. Therefore, obtaining a stable digital key (SDK) from the PUF is crucial. A fuzzy extractor (FE) is an algorithm

TABLE II
NOTATIONS GUIDE

| Notation | Description |
|---|---|
| $\mathcal{A}$ | Adversary |
| $\|, \oplus$ | Concatenation, XOR |
| $(R_i, C_i)$ | (challenge, response) pair |
| $PUF(\cdot)$ | Physical unclonable function |
| $h(\cdot)$ | Collision-resistant one-way cryptographic hash function |
| $E_q(\alpha, \beta)$, $P$ | Elliptic curve, and its base point |
| $Gen(\cdot), RP, Rep(\cdot)$ | FE key generation algorithm, reproduction parameter, reproduction algorithm |
| $VO_i, PW_{VO_i}$ | $i$th vehicle owner, owner's password |
| $RA_i$ | $i$th trusted registration authority |
| $OBU_i, RSU_j, CS_l$ | $i$th on-board unit, $j$th roadside unit, $l$th CS |
| $(PB_{OBU_i}, PR_{OBU_i})$, $(PB_{RSU_j}, PR_{RSU_j})$, $(PB_{CS_l}, PR_{CS_l})$ | Public/private key pair of $i$th OBU, $j$th RSU, $l$th CS |
| $R_{\text{AKE1}}, R_{\text{AKE2}}$ | Random numbers utilized in the AKE phase |
| $EC_{\bar{k}}(\cdot)/DC_{\bar{k}}(\cdot)$ | Symmetric encryption/decryption using private-key $\bar{k}$ |
| $RN_1, M_k, SP$ | Random nonce, encrypted parameter, secret parameter |
| $TS_1, TS_2$ | Timestamps utilized in AKE phase |
| $f(\alpha, \beta)$ | "$w$-degree symmetric-bivariate polynomial with property $f(\alpha, \beta) = f(\beta, \alpha)$" |

that can produce stable cryptographic keys from noisy output of the $PUF(\cdot)$ [31]. FE consists of two algorithms, the generation algorithm and the reproduction algorithm, denoted as $Gen(\cdot)$ and $Rep(\cdot)$, respectively. Specifically:

$Gen(R) \rightarrow (SDK, RP)$: $Gen(\cdot)$ accepts $R$ as input and produces a key $SDK$ and a reproduction parameter $RP$.

$Rep(R', RP) \rightarrow (SDK')$: $Rep(\cdot)$ takes $R'$ and $RP$ as inputs and produces $SDK'$ as an output. The correctness of the output is based on the two samples $R$ and $R'$. If $SDK$ and $SDK'$ are sufficiently close, then $SDK' = SDK$. That is, $SDK$ can be accurately reproduced.

## IV. THE PROPOSED SCHEME

This section presents the proposed AAKE-BIVT based on the network model illustrated in Fig. 1. It is worth recapping that for each VANET application $VANET_i$, there exists an $RA_i$ to register network entities. Table II summarizes the notations and their descriptions utilized in AAKE-BIVT. In the following subsections, we detail the AAKE-BIVT phases.

### A. System Initialization Phase

The system initialization (SI) phase is responsible for selecting the associated system parameters, detailed as follows.

*1) SI-1:* With each VANET application $VANET_i$, a trusted $RA_i$ selects an elliptic curve i.e., $E_q(\alpha, \beta)$, of the form $y^2 = x^3 + \alpha x + \beta \pmod q$ over finite field $Z_q$, where $q$ is a large prime, and $\alpha, \beta \in Z_q^*$, with the condition $4\alpha^3 - 27\beta^2 \neq 0 \pmod q$, along with $\mathcal{O}$ as the point at infinity or zero point.[1]

[1]In prime field arithmetic, $Z_q^*$ is $Z_q$ with its zero point removed.

Then, $RA_i$ picks a generation point or base point $P$, such that $P \in E_q(\alpha, \beta)$, of order, say $n$, i.e., $n \cdot P = \mathcal{O}$, where $n \cdot P$ shows the ECC point multiplication and $n \in Z_q^*$ is also called the discrete logarithm to the base $P$.

*2) SI-2:* $RA_i$ picks a one-way collision-resistant cryptographic hash function $h(\cdot)$. For instance, SHA-256 $h(\cdot)$ can be considered for providing sufficient security which gives a message digest of 256-bit.

*3) SI-3:* $RA_i$ picks a "$w$-degree symmetric-bivariate polynomial of the form $f(\alpha, \beta) = \sum_{k=0}^{w} \sum_{l=0}^{w} y_{kl} \alpha^k \beta^l \pmod q$ over finite field $Z_q$, where the coefficients $y_{kl} \in Z_q$, with the property that $f(\alpha, \beta) = f(\beta, \alpha)$". The degree $w$ of $f(\alpha, \beta)$ is chosen such that $w \gg$ the number of RSUs and CSs and the $w$-collision resistant properties are fulfilled.

It is worth mentioning that $\{E_q(\alpha, \beta), P, h(\cdot), f(\alpha, \beta)\}$ are securely shared among all the others RAs in the system.

### B. Registration Phase

The registration process of the individual network entities, such as vehicle owners (VOs), RSUs and CSs, with $RA_i$ for VANET application $VANET_i$ is elaborated below.

*1) Vehicle Owner Registration Phase:* Before accessing the online smart application (SA) $SA_\iota$, the VO $VO_j$ must register the OBU $OBU_j$ with the associated $RA_i$ offline by forwarding the vehicle documents and identity proof. The VO registration (VOR) phase is described below.

*a) Step VRP-1:* To register $OBU_j$, $VO_j$ forwards request to $RA_i$. $RA_i$ picks a unique challenge parameter $C_j$ and transmits it to $OBU_j$ via a secure channel. $OBU_j$ computes the response parameter $R_j = PUF(C_j)$, and further generates stable digital key $SDK_{OBU_j}$ and reproduction parameter $RP_{OBU_j}$ as $(SDK_{OBU_j}, RP_{OBU_j}) = Gen(R_j)$. Moreover, $OBU_j$ calculates identity $ID_{OBU_j}$ of $OBU_j$ as $ID_{OBU_j} = h(SDK_{OBU_j})$, and generates a random nonce $RN_j$. Next, $VO_j$ chooses the password $PW_{VO_j}$ and computes $RPW_{VO_j} = h(PW_{VO_j} \| RN_j)$. Then $\{ID_{OBU_j}, RP_{OBU_j}, RPW_{VO_j}\}$ are forwarded to $RA_i$ via a secure channel.

*b) Step VRP-2:* After receiving $\{ID_{OBU_j}, RP_{OBU_j}, RPW_{VO_j}\}$, $RA_i$ picks a private key $PR_{OBU_j} \in Z_q^*$ and calculates the public key $PB_{OBU_j}$ as $PB_{OBU_j} = PR_{OBU_j} \cdot P$. Next, $RA_i$ computes $Q_j = PR_{OBU_j} \oplus h(RPW_{VO_j} \| ID_{OBU_j})$ and $W_j = h(RPW_{VO_j} \| PR_{OBU_j})$, and prepares a smart card $SC = \{C_j, RP_{OBU_j}, Q_j, W_j\}$ for $VO_j$. $RA_i$ stores $C_j$ and $RP_{OBU_j}$ in its database and delivers $SC$ on mailing address.

*c) Step VRP-3:* After receiving $SC$, $VO_j$ computes $A_j = RN_j \oplus h(PW_{VO_j} \| ID_{OBU_j})$ and stores $\{A_j\}$ in $SC$.

It is worth noting that $VO_j$ needs to remember $PW_{VO_j}$ in order to access the smart transportation service $SA_\iota$.

*2) CS Registration:* Before deploying CS $CS_l$, the relevant $RA_i$ executes the following cloud service registration procedure (CRP).

*a) Step CRP-1:* $RA_i$ selects a unique real identity $ID_{CS_l}$ and a random temporary identity $TID_{CS_l}$, and calculates a pesudo-identity as $PID_{CS_l} = h(ID_{CS_l} \| PR_{RA_i} \| RT_{CS_l})$, where $RT_{CS_l}$ and $PR_{RA_i}$ are the time of $CS_l$ registration and private key of $RA_i$, respectively. $RA_i$ also selects a private key $PR_{CS_l}$ and calculates the public key as $PB_{CS_l} = PR_{CS_l} \cdot P$.

*b) Step CRP-2:* Pairwise secret keys among CSs are established using a key distribution technique based on symmetric-bivariate polynomial [32], (see Subsection IV-E). To accomplish this goal, $RA_i$ selects a "$w$-degree symmetric-bivariate polynomial of the form $f(\alpha,\beta)=\sum_{k=0}^{w}\sum_{j=0}^{w} y_{kj}\alpha^k\beta^j \pmod q$over finite field$Z_q$, with the property $f(\alpha,\beta)=f(\beta,\alpha)$", where the coefficients $y_{kj}\in Z_q$, and it further calculate a polynomial share for $CS_l$ as $f(PID_{CS_l},\beta)=\sum_{k=0}^{w}\sum_{j=0}^{w} y_{kj}PID_{CS_l}^k\beta^j \pmod q$, which fabricates a $w$-degree symmetric-univariate polynomial.

*c) Step CRP-3:* Finally, $RA_i$ loads the parameters $\{ID_{CS_l}, PR_{CS_l}, (TID_{CS_l}, PID_{CS_l}), f(PID_{CS_l},\beta)\}$ in $CS_l$. Moreover, $RA_i$ also stores $\{(TID_{CS_k}, PID_{CS_k}) \mid k\neq l, k=1,2,\cdots,n_{\rm CS}\}$ in $CS_l$ corresponding to all other CSs $CS_k$, and the parameter $PB_{CS_l}$ is published publicly.

*3) RSU Registration Phase:* The RSU registration procedure (RRP) is as follows.

*a) Step RRP-1:* To register $RSU_k$, a request is forwarded to $RA_i$. Then, $RA_i$ selects a unique challenge parameter $C_k$ and sends it to $RSU_k$ via a secure private channel. $RSU_k$ computes the response parameter as $R_k=PUF(C_k)$. Additionally, $RSU_k$ produces reproduction parameter $RP_{RSU_k}$ and stable digital key $SDK_{RSU_k}$ as $(SDK_{RSU_k}, RP_{RSU_k})=Gen(R_k)$. Moreover, $RSU_k$ computes identity as $ID_{RSU_k}=Z_0^a\oplus Z_0^b$, where $Z_0=h(SDK_{RSU_k})$, and $Z_0^a$ and $Z_0^b$ are extracted by splitting the parameter $Z_0$. Next, $ID_{RSU_k}$ and $R_k$ are sent to $RA_i$ using a secure private channel.

*b) Step RRP-2:* After acquiring $\{ID_{RSU_k}, R_k\}$, $RA_i$ selects a unique private key $PR_{RSU_k}$, random nonce $RN_1$ and secret parameter $SP$, to compute $PB_{RSU_k}=PR_{RSU_k}\cdot P$, $B_k=PR_{RSU_k}\oplus h(RN_1\parallel ID_{RSU_k})$, $SID_k=h(ID_{RSU_k})$, $\bar{k}=h(ID_{RSU_k}\parallel PR_{CS_m})$, and $M_k=EC_{\bar{k}}(SP\parallel C_k\parallel R_k)$, where $PB_{RSU_k}, SID_k, \bar{k}$, and $M_k$ are the public key, searching identity, symmetric key, and encrypted parameter, respectively. $RA_i$ stores $\{SID_k, M_k\}$ in $CS_m$ and forwards $\{B_k, RN_1, PB_{RSU_k}, SP\}$ to $RSU_k$ using a secure private channel.

*c) Step RRP-3:* After obtaining $\{B_k, RN_1, PB_{RSU_k}, SP\}$, $RSU_k$ computes $X_1=h(SP\parallel ID_{RSU_k})$, and $X_2=SP\oplus SDK_{RSU_k}$. Finally, $RSU_k$ stores $\{C_i, RP_{RSU_k}, B_k, RN_1, X_1, X_2, PB_{RSU_k}\}$, and $PB_{RSU_k}$ is published publicly.

### C. Vehicle User Login Phase

After successfully enrolling, $VO_j$ acquires a smart card $SC$ from the corresponding $RA_i$, which is used for logging into $OBU_j$ locally and accessing smart transportation services. The vehicle user login procedure is as follows.

*1) Step ULP-1:* $VO_j$ inserts smart card $SC$ into $OBU_j$, and enters the password, denoted as $PW_{VO_j}^l$, into $SA_\iota$.

*2) Step ULP-2:* $SA_\iota$ retrieves $\{C_j, RP_{OBU_j}\}$ from $SC$ and computes the challenge parameter $R_j=PUF(C_j)$, and further computes stable digital key $SDK_{OBU_j}$ for $OBU_j$ using fuzzy extractor $SDK_{OBU_j}=Rep(R_j, RP_{OBU_j})$. Next, it computes $ID_{OBU_j}=h(SDK_{OBU_j})$, $RN_j=A_j\oplus h(PW_{VO_j}^l\parallel ID_{OBU_j})$, $RPW_{VO_j}^l=h(PW_{VO_j}^l\parallel RN_j)$, $PR_{OBU_j}=Q_j\oplus h(RPW_{VO_j}\parallel ID_{OBU_j})$, and $W_j'=h(RPW_{VO_j}^l\parallel PR_{OBU_j})$. Finally, it checks if $W_j'\stackrel{?}{=}W_j$ holds. If so, $VO_j$ is successfully logged into $OBU_j$.

### D. Authenticated Key Exchange Phase

This subsection details the devised AKE schemes for the three different cases: 1) between an OBU (vehicle) and its neighboring OBU (vehicle), 2) between CH and RSU, and 3) between RSU and CS.

*1) AKE Between Vehicles:* Both the neighboring vehicles, $OBU_j$ and $OBU_{j\prime}$, must have login with the assistance of $SA_\iota$ and $SA_{\iota\prime}$, respectively, as elaborated in Subsection IV-C. The process of AKE between vehicles is as follows.

*a) Step KEV2V-1:* $OBU_j$ generates random nonce $R_{\rm AKE1}$, and picks current timestamp $TS_1$. Then it calculates $SSC=PR_{OBU_i}\cdot PB_{RSU_j}$, $ASC=R_{\rm AKE1}\cdot P$, $BSC=R_{\rm AKE1}\cdot PB_{RSU_j}$, $M_1=ASC\oplus PB_{OBU_i}$, and $Auth_1=h(ASC\parallel BSC\parallel TS_1)$. Next $OBU_j$ constructs message $msg_{VV_1}=\{M_1, Auth_1, BSC, TS_1\}$ and sends it to $OBU_{j\prime}$ via insecure channel.

*b) Step KEV2V-2:* After acquiring $msg_{VV_1}$ at time $TS_1'$, $OBU_{j\prime}$ verifies the condition $|TS_1-TS_1'|<\Delta T$? If so, $OBU_{j\prime}$ computes $ASC=BSC\cdot PR_{OBU_{j\prime}}^{-1}$, $PB_{OBU_j}=ASC\oplus M_1$, $SSC_2=PR_{OBU_{j\prime}}\cdot PB_{OBU_j}$, and $Auth_2=h(ASC\parallel BSC\parallel TS_1)$. Next $OBU_{j\prime}$ checks if $Auth_2\stackrel{?}{=}Auth_1$ holds. If true, $OBU_{j\prime}$ generates a nonce $R_{\rm AKE2}$, and picks current timestamp $TS_2$. Then, it computes $M_2=ASC\oplus R_{\rm AKE2}$, and also computes session key $SK_{OBU_{j\prime},OBU_j}$ and session key verifier $Auth_3$ as $SK_{OBU_{j\prime},OBU_j}=h(ASC\parallel R_{\rm AKE2}\parallel SSC_2\parallel TS_1\parallel TS_2)$, and $Auth_3=h(R_{\rm AKE2}\parallel SK_{OBU_{j\prime},OBU_j}\parallel TS_2)$, respectively. $OBU_{j\prime}$ constructs message $msg_{VV_2}=\{M_2, Auth_3, TS_2\}$ and sends it to $OBU_j$ using insecure channel.

*c) Step KEV2V-3:* After acquiring $msg_{VV_2}$ from $OBU_{j\prime}$, $OBU_j$ checks whether the current timestamp $TS_2'$ satisfies $|TS_2-TS_2'|<\Delta T$. If this condition holds, $OBU_j$ extracts $R_{\rm AKE2}$ from $R_{\rm AKE2}=M_2\oplus ASC$. Next, $OBU_j$ computes session key $SK_{OBU_j,OBU_{j\prime}}$ and session key verifier $Auth_4$ as $SK_{OBU_j,OBU_{j\prime}}=h(ASC\parallel R_{\rm AKE2}\parallel SSC\parallel TS_1\parallel TS_2)$, and $Auth_4=h(R_{\rm AKE2}\parallel SK_{OBU_j,OBU_{j\prime}}\parallel TS_2)$, respectively. $OBU_j$ checks if $Auth_4\stackrel{?}{=}Auth_3$ holds, then it accepts and stores $SK_{OBU_j,OBU_j}(=SK_{OBU_{j\prime},OBU_j})$ as SK.

This authentication and key exchange phase between two neighboring vehicles is summarized in Fig. 2.

*2) AKE Between CH and RSU:* The process of AKE between cluster head CH (e.g., $OBU_i$) and RSU $RSU_j$ is detailed as follows.

*a) Step KEV2R-1:* $CH$ ($OBU_i$) generates random nonce $R_{\rm AKE1}$ and picks current timestamp $TS_1$. Next it computes $SSC=PR_{OBU_i}\cdot PB_{RSU_j}$, $ASC=R_{\rm AKE1}\cdot P$, $BSC=R_{\rm AKE1}\cdot PB_{RSU_j}$, $M_1=ASC\oplus PB_{OBU_i}$, and $Auth_1=h(ASC\parallel BSC\parallel TS_1)$. Then $CH$ constructs message $msg_{VR_1}=\{M_1, Auth_1, BSC, TS_1\}$ and transmits it to $RSU_j$ through public channel.

*b) Step KEV2R-2:* After receiving $msg_{VR_1}$ from $CH$, $RSU_j$ checks whether the current timestamp $TS_1'$ satisfies $|TS_1-TS_1'|<\Delta T$. If this condition holds, $RSU_j$ retrieves parameters $C_i$ and $RP_{RSU_j}$, and computes $R_i=PUF(C_i)$,

| **Vehicle** $OBU_j$ | **Vehicle** $OBU_{j\prime}$ |
|---|---|
| Known parameters:$\{A_j, C_j, RP_{OBU_j}, Q_j, W_j\}$; Input: $PW^l_{VO_j}$ | Known parameters:$\{A_{j\prime}, C_{j\prime}, RP_{OBU_{j\prime}}, Q_{j\prime}, W_{j\prime}\}$; Input: $PW^l_{VO_{j\prime}}$ |
| Retrieve: $C_j, RP_{OBU_j}$;<br>Compute: $R_j = PUF(C_j)$, $SDK_{OBU_j} = Rep(R_j, RP_{OBU_j})$, $ID_{OBU_j} = h(SDK_{OBU_j})$, $RN_j = A_j \oplus h(PW^l_{VO_j} \| ID_{OBU_j})$, $RPW^l_{VO_j} = h(PW^l_{VO_j} \| RN_j)$, $PR_{OBU_j} = Q_j \oplus h(RPW_{VO_j} \| ID_{OBU_j})$, $W'_j = h(RPW^l_{VO_j} \| PR_{OBU_j})$;<br>Check if $W'_j \stackrel{?}{=} W_j$ holds: Vehicle driver is successfully login. | Retrieve: $C_{j\prime}, RP_{OBU_{j\prime}}$;<br>Compute: $R_{j\prime} = PUF(C_{j\prime})$, $SDK_{OBU_{j\prime}} = Rep(R_{j\prime}, RP_{OBU_{j\prime}})$, $ID_{OBU_{j\prime}} = h(SDK_{OBU_{j\prime}})$, $RN_{j\prime} = A_{j\prime} \oplus h(PW^l_{VO_{j\prime}} \| ID_{OBU_{j\prime}})$, $RPW^l_{VO_{j\prime}} = h(PW^l_{VO_{j\prime}} \| RN_{j\prime})$, $PR_{OBU_{j\prime}} = Q_{j\prime} \oplus h(RPW^l_{VO_{j\prime}} \| ID_{OBU_{j\prime}})$, $W'_{j\prime} = h(RPW^l_{VO_{j\prime}} \| PR_{OBU_{j\prime}})$;<br>Check if $W'_{j\prime} \stackrel{?}{=} W_{j\prime}$ holds: Vehicle driver is successfully login. |
| Mutual authentication and key exchange scheme | |
| Generate: random nonce $R_{\text{AKE1}}$;<br>Pick: current timestamp $TS_1$;<br>Compute: $SSC = PR_{OBU_j} \cdot PB_{OBU_{j\prime}}$, $ASC = R_{\text{AKE1}} \cdot P$, $BSC = R_{\text{AKE1}} \cdot PB_{OBU_{j\prime}}$, $M_1 = ASC \oplus PB_{OBU_j}$, $Auth_1 = h(ASC \| BSC \| TS_1)$;<br>$\xrightarrow[(OBU_j \to OBU_{j\prime})]{msg_{VV_1}:\{M_1, Auth_1, BSC, TS_1\}}$. | Check if $\|TS_1 - TS'_1\| < \Delta T$? If true, compute: $ASC = BSC \cdot PR^{-1}_{OBU_{j\prime}}$, $PB_{OBU_j} = ASC \oplus M_1$, $SSC_2 = PR_{OBU_{j\prime}} \cdot PB_{OBU_j}$, $Auth_2 = h(ASC \| BSC \| TS_1)$;<br>Check if $Auth_2 \stackrel{?}{=} Auth_1$ holds; If so, generate: $TS_2$ and $R_{\text{AKE2}}$;<br>Compute: $M_2 = ASC \oplus R_{\text{AKE2}}$, $SK_{OBU_{j\prime},OBU_j} = h(ASC \| R_{\text{AKE2}} \| SSC_2 \| TS_1 \| TS_2)$,<br>$Auth_3 = h(R_{\text{AKE2}} \| SK_{OBU_{j\prime},OBU_j} \| TS_2)$;<br>$\xleftarrow[(OBU_{j\prime} \to OBU_j)]{msg_{VV_2}:\{M_2, Auth_3, TS_2\}}$. |
| Check if $\|TS_2 - TS'_2\| < \Delta T$? If true, extract: $R_{\text{AKE2}}$ from $R_{\text{AKE2}} = M_2 \oplus ASC$;<br>Compute: $SK_{OBU_j,OBU_{j\prime}} = h(ASC \| R_{\text{AKE2}} \| SSC \| TS_1 \| TS_2)$, $Auth_4 = h(R_{\text{AKE2}} \| SK_{OBU_j,OBU_{j\prime}} \| TS_2)$;<br>Check if $Auth_4 \stackrel{?}{=} Auth_3$ holds;<br>Store $SK_{OBU_j,OBU_{j\prime}} (= SK_{OBU_{j\prime},OBU_j})$ as SK. | |
| $SK_{OBU_j,OBU_{j\prime}} (= SK_{OBU_{j\prime},OBU_j}) = h((R_{\text{AKE1}} \cdot P) \| R_{\text{AKE2}} \| (PR_{OBU_j} \cdot PR_{OBU_{j\prime}} \cdot P) \| TS_1 \| TS_2)$ | |

Fig. 2. Authentication and key exchange phase between two neighboring vehicles $OBU_j$ and $OBU_{j\prime}$.

$SDK_{RSU_j} = Rep(R_i, RP_{RSU_j})$, $Z_0 = h(SDK_{RSU_j})$, $ID_{RSU_j} = Z^a_0 \oplus Z^b_0$, and $PR_{RSU_j} = B_j \oplus h(RN_1 \| ID_{RSU_j})$. $RSU_j$ further calculates $ASC = BSC \cdot PR^{-1}_{RSU_j}$, $PB_{OBU_i} = ASC \oplus M_1$, $SSC_2 = PR_{RSU_j} \cdot PB_{OBU_i}$, and $Auth_2 = h(ASC \| BSC \| TS_1)$. Next it checks if $Auth_2 \stackrel{?}{=} Auth_1$ holds. If true, $RSU_j$ generates a nonce $R_{\text{AKE2}}$ and picks current timestamp $TS_2$. Then it computes $M_2 = ASC \oplus R_{\text{AKE2}}$, and further calculates session key $SK_{RSU_j,OBU_i}$ and session key verifier $Auth_3$ as $SK_{RSU_j,OBU_i} = h(ASC \| R_{\text{AKE2}} \| SSC_2 \| TS_1 \| TS_2)$ and $Auth_3 = h(R_{\text{AKE2}} \| SK_{RSU_j,OBU_i} \| TS_2)$, respectively. $RSU_j$ fabricates message $msg_{VR_2} = \{M_2, Auth_3, TS_2\}$, and then transmits it to $OBU_i$ using an insecure channel.

*c) Step KEV2R-3:* After acquiring $msg_{VR_2}$ from $RSU_j$, $CH$ $(OBU_i)$ checks whether the current timestamp $TS'_2$ satisfies $|TS_2 - TS'_2| < \Delta T$. If this condition holds, $CH$ extracts $R_{\text{AKE2}}$ from $R_{\text{AKE2}} = M_2 \oplus ASC$. Next $CH$ computes session key $SK_{OBU_i,RSU_j}$ and session key verifier $Auth_4$ as $SK_{OBU_i,RSU_j} = h(ASC \| R_{\text{AKE2}} \| SSC \| TS_1 \| TS_2)$ and $Auth_4 = h(R_{\text{AKE2}} \| SK_{OBU_i,RSU_j} \| TS_2)$, respectively. It then checks if $Auth_4 \stackrel{?}{=} Auth_3$ holds, then accepts and stores $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i})$ as SK.

The authentication and key exchange phase between CH (OBS) and RSU is illustrated in Fig. 3.

*3) AKE Between RSU and CS:* The proposed AKE scheme between $RSU_j$ and $CS_m$ is detailed as follows.

*a) Step KER2C-1:* $RSU_j$ retrieves $C_i$ and computes $R_i = PUF(C_i)$, $SDK_{RSU_j} = Rep(R_i, RP_{RSU_j})$, $Z_0 = h(SDK_{RSU_j})$, $ID_{RSU_j} = Z^a_0 \oplus Z^b_0$, $SP' = X_2 \oplus SDK_{RSU_j}$, and $X'_1 = h(SP' \| ID_{RSU_j})$. Next, $RSU_j$ checks if $X_1 \stackrel{?}{=} X'_1$ holds. If so, it generates $R_{\text{AKE1}}$ and picks $TS_1$, and computes $PR_{RSU_j} = B_k \oplus h(RN_1 \| ID_{RSU_j})$, $SSH = PR_{RSU_j} \cdot PB_{CS_m}$, $ASC = R_{\text{AKE1}} \cdot P$, $BSC = R_{\text{AKE1}} \cdot PB_{CS_m}$, $Z_3 = ASC \oplus PB_{RSU_j}$, $Z_4 = (ID_{RSU_j} \| RP_{RSU_j}) \oplus h(SSH \| TS_1)$, and $Z_5 = h(ASC \| BSC \| ID_{RSU_j} \| RP_{RSU_j} \| SP \| TS_1)$. $RSU_j$ forwards message $msg_{RC_1} = \{Z_3, BSC, Z_4, Z_5, TS_1\}$ to $CS_m$ using an insecure channel.

*b) Step KER2C-2:* After reception of $msg_{RC_1}$ at $TS'_1$, $CS_m$ checks $|TS_1 - TS'_1| < \Delta T$? If so, it computes $ASC = BSC \cdot PR^{-1}_{CS_m}$, $PB_{RSU_j} = ASC \oplus Z_3$, $SSH_2 = PR_{CS_m} \cdot PB_{RSU_j}$, and extracts $ID_{RSU_j}$ and $RP_{RSU_j}$ from $(ID_{RSU_j} \| RP_{RSU_j}) = Z_4 \oplus h(SSH_2 \| TS_1)$. To extract the secret credentials ($SID_j$ and $M_j$) of $RSU_j$ stored in its database for authentication of $RSU_j$, $CS_m$ computes $SID_j = h(ID_{RSU_j})$, and through $SID_j$, it retrieves $M_j$ from its database. Then $CS_m$ computes $K_2 = h(ID_{RSU_j} \| PR_{CS_m})$, $(SP \| R_i \| C_i) = DC_{K_2}(M_j)$, and $Z'_5 = h(ASC \| BSC \| ID_{RSU_j} \| RP_{RSU_j} \| SP \| TS_1)$. Next $CS_m$ checks $Z_5 \stackrel{?}{=} Z'_5$. If so, it generates $R_{\text{AKE2}}$ and picks $TS_2$. Then $CS_m$ computes $SDK^a_{RSU_j} = Rep(R_i, RP_{RSU_j})$ and $Z_6 = ASC \oplus R_{\text{AKE2}}$. Moreover, $CS_m$ calculates session key $SK_{CS_m,RSU_j} = h(ASC \| R_{\text{AKE2}} \| SSH_2 \| SDK^a_{RSU_j} \| SP \| TS_1 \| TS_2)$, and also computes session key verifier $Z_7 = h(SK_{CS_m,RSU_j} \| R_{\text{AKE2}} \| TS_2)$. Finally $CS_m$ forwards the message $msg_{RC_2} = \{Z_6, Z_7, TS_2\}$ to $RSU_j$ via an insecure channel.

*c) Step KER2C-3:* After receiving $msg_{RC_2}$ from $CS_m$, $RSU_j$ checks whether the current timestamp $TS'_2$ satisfies $|TS_2 - TS'_2| < \Delta T$. If this condition holds, $RSU_j$ extracts $R_{\text{AKE2}}$ from $R_{\text{AKE2}} = Z_6 \oplus ASC$. Next $RSU_j$ calculates

| **Cluster head** $OBU_i$ | **Roadside Unit** $RSU_j$ |
|---|---|
| $\{A_i, C_i, RP_{OBU_i}, Q_i, W_i\}$ | $\{C_i, RP_{RSU_j}, B_j, RN_1, X_1, X_2, PB_{RSU_j}\}$ |
| Generate: random nonce $R_{\text{AKE1}}$;<br>Pick: current timestamp $TS_1$;<br>Compute: $SSC = PR_{OBU_i} \cdot PB_{RSU_j}$, $ASC = R_{\text{AKE1}} \cdot P$, $BSC = R_{\text{AKE1}} \cdot PB_{RSU_j}$, $M_1 = ASC \oplus PB_{OBU_i}$, $Auth_1 = h(ASC \parallel BSC \parallel TS_1)$;<br>$\xrightarrow[(OBU_i \rightarrow RSU_j)]{msg_{VR_1}:\{M_1, Auth_1, BSC, TS_1\}}$. | Check if $\lvert TS_1 - TS_1'\rvert < \Delta T$? If so, retrieve: $C_i$ and $RP_{RSU_j}$;<br>Compute: $R_i = PUF(C_i)$, $SDK_{RSU_j} = Rep(R_i, RP_{RSU_j})$, $Z_0 = h(SDK_{RSU_j})$, $ID_{RSU_j} = Z_0^a \oplus Z_0^b$, $PR_{RSU_j} = B_j \oplus h(RN_1 \parallel ID_{RSU_j})$, $ASC = BSC \cdot PR_{RSU_j}^{-1}$, $PB_{OBU_i} = ASC \oplus M_1$, $SSC_2 = PR_{RSU_j} \cdot PB_{OBU_i}$, $Auth_2 = h(ASC \parallel BSC \parallel TS_1)$;<br>Check if $Auth_2 \stackrel{?}{=} Auth_1$ holds;<br>If so, generate: random nonce $R_{\text{AKE2}}$ and pick: $TS_2$;<br>Compute: $M_2 = ASC \oplus R_{\text{AKE2}}$, $SK_{RSU_j,OBU_i} = h(ASC \parallel R_{\text{AKE2}} \parallel SSC_2 \parallel TS_1 \parallel TS_2)$, $Auth_3 = h(R_{\text{AKE2}} \parallel SK_{RSU_j,OBU_i} \parallel TS_2)$;<br>$\xleftarrow[(RSU_j \rightarrow OBU_i)]{msg_{VR_2}:\{M_2, Auth_3, TS_2\}}$. |
| Check if $\lvert TS_2 - TS_2'\rvert < \Delta T$? If so, extract: $R_{\text{AKE2}}$ from $R_{\text{AKE2}} = M_2 \oplus ASC$;<br>Compute: $SK_{OBU_i,RSU_j} = h(ASC \parallel R_{\text{AKE2}} \parallel SSC \parallel TS_1 \parallel TS_2)$, $Auth_4 = h(R_{\text{AKE2}} \parallel SK_{OBU_i,RSU_j} \parallel TS_2)$;<br>Check if $Auth_4 \stackrel{?}{=} Auth_3$ holds;<br>Store: $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i})$ as SK. | |
| $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{OBU_i} \cdot PR_{RSU_j} \cdot G) \parallel TS_1 \parallel TS_2)$. | |

Fig. 3. Authentication and key exchange phase between CH $OBU_i$ and RSU $RSU_j$.

| **Roadside Unit** $RSU_j$ | **Cloud Server** $CS_m$ |
|---|---|
| $\{C_i, RP_{RSU_j}, B_j, RN_1, X_1, X_2, PB_{RSU_j}\}$ | $\{SID_j, M_j\}$ |
| Retrieve: $C_i$;<br>Compute: $R_i = PUF(C_i)$, $SDK_{RSU_j} = Rep(R_i, RP_{RSU_j})$, $Z_0 = h(SDK_{RSU_j})$, $ID_{RSU_j} = Z_0^a \oplus Z_0^b$, $SP' = X_2 \oplus SDK_{RSU_j}$, $X_1' = h(SP' \parallel ID_{RSU_j})$;<br>Check if $X_1 \stackrel{?}{=} X_1'$ holds; If so, generate: random nonce $R_{\text{AKE1}}$;<br>Pick: current timestamp $TS_1$;<br>Compute: $PR_{RSU_j} = B_j \oplus h(RN_1 \parallel ID_{RSU_j})$, $SSH = PR_{RSU_j} \cdot PB_{CS_m}$, $ASC = R_{\text{AKE1}} \cdot P$, $BSC = R_{\text{AKE1}} \cdot PB_{CS_m}$, $Z_3 = ASC \oplus PB_{RSU_j}$, $Z_4 = (ID_{RSU_j} \parallel RP_{RSU_j}) \oplus h(SSH \parallel TS_1)$, $Z_5 = h(ASC \parallel BSC \parallel ID_{RSU_j} \parallel RP_{RSU_j} \parallel SP \parallel TS_1)$;<br>$\xrightarrow[(RSU_j \rightarrow CS_m)]{msg_{RC_1}:\{Z_3,\ BSC,\ Z_4,\ Z_5,\ TS_1\}}$. | Check if $\lvert TS_1 - TS_1'\rvert < \Delta T$? If so, compute: $ASC = BSC \cdot PR_{CS_m}^{-1}$, $PB_{RSU_j} = ASC \oplus Z_3$, $SSH_2 = PR_{CS_m} \cdot PB_{RSU_j}$;<br>Extract: $ID_{RSU_j}$ and $RP_{RSU_j}$ from $(ID_{RSU_j} \parallel RP_{RSU_j}) = Z_4 \oplus h(SSH_2 \parallel TS_1)$;<br>Compute: $SID_j = h(ID_{RSU_j})$, and through $SID_j$, retrieve: $M_j$;<br>Calculate: $K_2 = h(ID_{RSU_j} \parallel PR_{CS_m})$, $(SP \parallel R_i \parallel C_i) = DC_{K_2}(M_j)$, $Z_5' = h(ASC \parallel BSC \parallel ID_{RSU_j} \parallel RP_{RSU_j} \parallel SP \parallel TS_1)$;<br>Check if $Z_5 \stackrel{?}{=} Z_5'$ holds; If so, generate: $R_{\text{AKE2}}$; Pick: $TS_2$;<br>Compute: $SDK_{RSU_j}^a = Rep(R_i, RP_{RSU_j})$, $Z_6 = ASC \oplus R_{\text{AKE2}}$, $SK_{CS_m,RSU_j} = h(ASC \parallel R_{\text{AKE2}} \parallel SSH_2 \parallel SDK_{RSU_j}^a \parallel SP \parallel TS_1 \parallel TS_2)$, $Z_7 = h(SK_{CS_m,RSU_j} \parallel R_{\text{AKE2}} \parallel TS_2)$;<br>$\xleftarrow[(CS_m \rightarrow RSU_j)]{msg_{RC_2}:\{Z_6,\ Z_7,\ TS_2\}}$. |
| Check if $\lvert TS_2 - TS_2'\rvert < \Delta T$? If so, extract: $R_{\text{AKE2}}$ from $R_{\text{AKE2}} = Z_6 \oplus ASC$;<br>Compute: $SK_{RSU_j,CS_m} = h(ASC \parallel R_{\text{AKE2}} \parallel SSH \parallel SDK_{RSU_j} \parallel SP \parallel TS_1 \parallel TS_2)$, $Z_7' = h(SK_{RSU_j,CS_m} \parallel R_{\text{AKE2}} \parallel TS_2)$;<br>Check: $Z_7 \stackrel{?}{=} Z_7'$ holds;<br>Store: $SK_{RSU_j,CS_m} (= SK_{CS_m,RSU_j})$ as SK. | |
| $SK_{RSU_j,CS_m} (= SK_{CS_m,RSU_j}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{RSU_j} \cdot PR_{CS_m} \cdot G) \parallel SDK_{RSU_j} \parallel SP \parallel TS_1 \parallel TS_2)$. | |

Fig. 4. Authentication and key agreement phase between $RSU_j$ and $CS_m$.

session key $SK_{RSU_j,CS_m} = h(ASC \parallel R_{\text{AKE2}} \parallel SSH \parallel SDK_{RSU_j} \parallel SP \parallel TS_1 \parallel TS_2)$ and session key verifier $Z_7' = h(SK_{RSU_j,CS_m} \parallel R_{\text{AKE2}} \parallel TS_2)$. It then checks if $Z_7 \stackrel{?}{=} Z_7'$ holds, Then it stores $SK_{RSU_j,CS_m} (= SK_{CS_m,RSU_j})$ as SK.

The AKE between $RSU_j$ and $CS_m$ is illustrated in Fig. 4.

### *E. Key Management Phase*

The key management between CS $CSl$ and CS $CSj$ in the CSN is described in this subsection. Both $CS_l$ and $CS_j$ generate a secret key for future secure communications at the end of successful key management.

*1) Step 1:* $CS_l$ generates a random number $rn_{CS_l} \in Z_q^*$ and picks current timestamp $TS_{CS_l}$. Next, it calculates $f(PID_{CS_l}, PID_{CS_j})$ by taking the polynomial share $f(PID_{CS_l}, y)$ using $PID_{CS_j}$ of $CS_j$ as well as computes $A_{CS_l} = h(rn_{CS_l} \parallel T_{CS_l} \parallel PR_{CS_l}) \oplus h(f(PID_{CS_l}, PID_{CS_j}) \parallel TS_{CS_l})$ and $Auth_{CS_l} = h(PID_{CS_l} \parallel PID_{CS_j} \parallel A_{CS_l} \parallel h(f(PID_{CS_l}, PID_{CS_j}) \parallel TS_{CS_l})$. Then $CS_l$ sends message

$msg_{CS_l} = \{TID_{CS_l}, A_{CS_l}, Auth_{CS_l}, TS_{CS_l}\}$ to $CS_j$ through public insecure channel.

*2) Step 2:* After receiving $msg_{CS_l}$ at $TS'_{CS_l}$, $CS_j$ verifies the validity of $msg_{CS_l}$ by checking if $|TS_{CS_l} - TS'_{CS_l}| < \Delta T$. If so, $CS_j$ retrieves $PID_{CS_l}$ corresponding to $TID_{CS_l}$ from $msg_{CS_l}$. Next $CS_j$ computes $h(rn_{CS_l} \parallel T_{CS_l} \parallel PR_{CS_l}) = A_{CS_l} \oplus h(f(PID_{CS_j}, PID_{CS_l}) \parallel TS_{CS_l})$ and $Auth_{CS_j} = h(PID_{CS_l} \parallel PID_{CS_j} \parallel A_{CS_l} \parallel h(f(PID_{CS_j}, PID_{CS_l}) \parallel TS_{CS_l})$, and verifies if $Auth_{CS_j} \stackrel{?}{=} Auth_{CS_j}$ holds. If so, $CS_j$ generates random number $rn_{CS_j}$ and selects timestamp $TS_{CS_j}$. Then $CS_j$ calculates $A_{CS_j} = h(rn_{CS_j} \parallel TS_{CS_j} \parallel PR_{CS_j}) \oplus h(f(PID_{CS_j}, PID_{CS_l}) \parallel TS_{CS_j})$, secret key shared with $CS_l$ $SK_{CS_j,CS_l} = h(h(rn_{CS_l} \parallel T_{CS_l} \parallel PR_{CS_l}) \parallel h(rn_{CS_j} \parallel TS_{CS_j} \parallel PR_{CS_j}) \parallel h(f(PID_{CS_j}, PID_{CS_l}))$ and secret key verifier $SKV_{CS_j,CS_l} = h(SK_{CS_j,CS_l} \parallel h(rn_{CS_j} \parallel T_{CS_j} \parallel PR_{CS_j}) \parallel T_{CS_j})$. Next $CS_j$ sends the response message $msg_{CS_j} = \{SK_{CS_j,CS_l}, A_{CS_j}, SKV_{CS_j,CS_l}, T_{CS_j}\}$ to $CS_l$ through public channel.

*3) Step 3:* When $CSl$ receives the response message $msg_{CS_j}$ at $TS'_{CS_j}$, $CS_l$ checks the freshness of the received $msg_{CS_j}$ by checking if $|TS_{CS_j} - TS'_{CS_j}| < \Delta T$. If so, $CS_l$ calculates $h(rn_{CS_j} \parallel TS_{CS_j} \parallel PR_{CS_j}) = A_{CS_j} \oplus h(f(PID_{CS_j}, PID_{CS_l}) \parallel TS_{CS_j})$, and the secret key shared with $CS_j$ $SK_{CS_l,CS_j} = h(h(rn_{CS_l} \parallel T_{CS_l} \parallel PR_{CS_l}) \parallel h(rn_{CS_j} \parallel TS_{CS_j} \parallel PR_{CS_j}) \parallel h(f(PID_{CS_l}, PID_{CS_j}))$. Then $CS_l$ computes the secret key verifier $SKV_{CS_l,CS_j} = h(SK_{CS_l,CS_j} \parallel h(rn_{CS_j} \parallel T_{CS_j} \parallel PR_{CS_j}) \parallel T_{CS_j})$, and checks if $SKV_{CS_l,CS_j} \stackrel{?}{=} SKV_{CS_j,CS_l}$ holds. If so, the received message $msg_{CS_j}$ is legitimate. Hence, both $CS_l$ and $CS_j$ share the same pairwise secret key $SK_{CS_l,CS_j} = SK_{CS_j,CS_l}$ and utilize it for secure communications.

### F. Dynamic Node Addition Phase

To deploy a new RSU $RSU^{new}$ in the smart transportation network under an existing CS $CS_m$, $RA_i$ performs the following steps.

*1) Step NAP-1:* A request is initiated by $RSU^{new}$ to $RA_i$. $RA_i$ selects a unique challenge parameter $C$ and forwards it to $RSU^{new}$ via a private channel. Then $RSU^{new}$ computes response parameter $R = PUF(C)$, and produces stable digital key $SDK_{RSU^{new}}$ and reproduction parameter $RP_{RSU^{new}}$ by $(SDK_{RSU^{new}}, RP_{RSU^{new}}) = Gen(R)$. Moreover, $RSU^{new}$ computes identity $ID_{RSU^{new}} = X_0^a \oplus X_0^b$, where $X_0^a$ and $X_0^b$ are extracted by splitting $X_0 = h(SDK_{RSU^{new}})$. Next, it dispatches $(ID_{RSU^{new}}, R)$ through private channel to $RA_i$.

*2) Step NAP-2:* After receiving $(ID_{RSU^{new}}, R)$, $RA_i$ picks unique private key $PR_{RSU^{new}}$, random nonce $RN_1$, and secret parameter $SP$. It then calculates public key $PB_{RSU^{new}} = PR_{RSU^{new}} \cdot P$, $B_k = PR_{RSU^{new}} \oplus h(RN_1 \parallel ID_{RSU^{new}})$, searching identity $SID_k = h(ID_{RSU^{new}})$, symmetric key $k = h(ID_{RSU^{new}} \parallel PR_{CS_m})$, and encrypted parameter with key as $k$ $M_k = EC_k(SP \parallel C \parallel R)$. $RA_i$ stores $\{SID_k, M_k\}$ in $CS_m$ and forwards $\{B_k, RN_1, PB_{RSU^{new}}, SP\}$ to $RSU^{new}$ via a secure private channel.

*3) Step NAP-3:* After obtaining $\{B_k, RN_1, PB_{RSU^{new}}, SP\}$ from $RA_i$, $RSU^{new}$ computes $X_1 = h(SP \parallel ID_{RSU^{new}})$, $X_2 = SP \oplus SDK_{RSU^{new}}$. Finally, $RSU^{new}$ stores the parameters $\{C, RP_{RSU^{new}}, B_k, RN_1, X_1, X_2, PB_{RSU^{new}}\}$, and the parameter $PB_{RSU^{new}}$ is published publicly.

Similarly, a new CS $CS^{new}$ can be registered in the existing CSN by $RA_i$, as explained in Subsection IV-B2 *CS Registration*, before its deployment.

### G. Password Updation Phase

When vehicle user $VO_i$ wants to reset password due to security reasons, it must put smart card $SC$ into the onboard unit $OBU_i$ and requests to run the smart application $SA_i$. Then the following required steps are executed.

*1) Step PU-1:* $SA_i$ prompts $VO_i$ to put the password. Then, $VO_i$ enters password $PW_{VO_i}^{old}$.

*2) Step PU-2:* $SA_i$ retrieves $C_i$ and $RP_{OBU_j}$ from $SC$, and computes challenge parameter $R_i = PUF(C_i)$ and stable digital key $SDK_{OBU_i} = Rep(R_i, RP_{OBU_i})$. Next, it computes $ID_{OBU_i} = h(SDK_{OBU_i})$, $RN_i = A_i \oplus h(PW_{VO_i}^{old} \parallel ID_{OBU_i})$, $RPW_{VO_i} = h(PW_{VO_i}^{old} \parallel RN_i)$, $PR_{OBU_i} = Q_i \oplus h(RPW_{VO_i} \parallel ID_{OBU_i})$, $W_i' = h(RPW_{VO_i} \parallel PR_{OBU_i})$, and check if $W_i' \stackrel{?}{=} W_i$ holds. If so, $VO_i$ is successfully login into $OBU_i$. Now $VO_i$ can reset the password.

*3) Step PU-3:* $VO_i$ enters new password $PW_{VO_i}^{new}$ into $SA_i$. $SA_i$ with the aid of $SC$ computes $RPW_{VO_i}^{new} = h(PW_{VO_i}^{new} \parallel RN_i)$, and it further calculates $Q_i^{new} = PR_{OBU_i} \oplus h(RPW_{VO_i}^{new} \parallel ID_{OBU_i})$, $W_i^{new} = h(RPW_{VO_i}^{new} \parallel PR_{OBU_i})$ and $A_i^{new} = RN_i \oplus h(PW_{VO_i}^{new} \parallel ID_{OBU_i})$.

*4) Step PU-4:* $SA_i$ keeps the updated parameters in SC as $SC = \{C_i, RP_{OBU_i}, Q_i^{new}, W_i^{new}, A_i^{new}\}$.

### H. Block Construction and Addition Phase

This subsection first details block construction by a CS, and then discusses the block verification and addition to the blockchain network via a consensus mechanism.

*1) Block Construction Phase:* In the devised AAKE-BIVT, the information is securely transmitted in each traffic vicinity $TV_i$ among the vehicle and CH via an established session key $SK_{OBU_i,OBU_j}$ between vehicle and CH during the AKE phase discussed in Subsection IV-D1. The collected information by the CH is then secretly transferred to RSU using the established session key $SK_{OBU_j,RSU_k}$ between the CH $OBU_j$ and its associated RSU $RSU_k$ in the traffic vicinity $TV_i$ during the AKE phase discussed in Subsection IV-D2. Next, $RSU_k$ forwards the collected information secretly using the established session key as discussed in Subsection IV-D3 to its associated CS $CS_m$.

$CS_m$ forms the transaction, encrypts it with its public key, and puts it into the global transaction pool $GTNX_{pl}$. When the number of transactions in $GTNX_{pl}$ hits a threshold value, a leader $CS_l$ is selected through the secure leader selection algorithm of [33] from the P2P CSN. The leader $CS_l$ produces a block, which has the structure as illustrated in Fig. 5.

*2) Block Verification and Addition Phase:* Once block $Block_k$ is constructed by leader $CS_l$, the voting-based consensus mechanism using Ripple protocol consensus algorithm (RPCA) [34] is executed in the AAKE-BIVT for the block

| **Block Header** | |
|---|---|
| Block Version | $BlkVer$ |
| Previous Block Hash | $PreBlkHash$ |
| Merkle Tree Root | $MRHash$ |
| Timestamp | $TS$ |
| Ower of Block | $OB$ |
| Public Key of Owner | $PB_{CS}$ |
| **Block Payload (Encrypted Transactions)** | |
| List of Encrypted Transactions | $EP_{PB_{CS}}(Tx_i)\|i = 1, 2, \cdots, n_t)$ |
| Current Block Hash | $CBHash$ |
| Signature on $CBHash$ | $SignCBHash$ |

Fig. 5. Structure of block $Block_k$.

verification and addition into the blockchain, this process is summarized in Algorithm 1.

## V. Security Analysis

### A. Informal Security Analysis

We use informal security analysis to reveal that our proposed AAKE-BIVT is resilient against various potential attacks.

*1) Replay Attack:* During the AKE phase between two neighboring OBUs, $OBU_i$ and $OBU_j$, presented in Subsection IV-D1, the transmitted messages $msg_{VV_1} = \{M_1,\ Auth_1,\ BDC, TS_1\}$ and $msg_{VV_2} = \{M_2,\ Auth_3,\ TS_2\}$ are transmitted over insecure public channels. Similarly, during the AKE phase between CH and associated RSU, $OBU_i$ and $RSU_k$, as stated in Subsection IV-D2, the messages $msg_{VR_1} = \{M_1,\ Auth_1,\ BDC, TS_1\}$ and $msg_{VR_2} = \{M_2,\ Auth_3,\ TS_2\}$ are communicated over insecure public channels. Likewise, at the AKE phase between RSU and its associated CS as mentioned in Subsection IV-D3, the communicated messages $msg_{RC_1} = \{Z_3,\ BDC,\ Z_4,\ Z_5,\ TS_1\}$ and $msg_{RC_2} = \{Z_6,\ Z_7,\ TS_2\}$ are sent over insecure public channels.

Due to employing the current timestamps and random numbers in fabricating the communicated messages. If an adversary $\mathcal{A}$ tries to reply the old messages, then it can be easily detected because the freshness of the messages is validated first at the receiving end. Thus, our devised AAKE-BIVT inherently resists the replay attack.

*2) MitM Attack:* Adversary $\mathcal{A}$ may intercept the communication channel between vehicles, $OBU_i$ and $OBU_j$. For example, $\mathcal{A}$ tries to eavesdrop on the AKE request message $msg_{VV_1}$ from the insecure public channel and strives to modify it in order to impersonate a legitimate party in the network. To achieve this goal, however, $\mathcal{A}$ has to choose the correct random nonce and timestamp. $\mathcal{A}$ also requires the secret key $PR_{OBU_i}$ of the $OBU_i$. Therefore, it is a computationally infeasible task for $\mathcal{A}$ to fabricate the message $msg_{VV_1}$. Likewise, $\mathcal{A}$ cannot fabricate the acknowledgment AKE message $msg_{VV_2}$ without the knowledge of secret credential $PR_{OBU_j}$. Similarly, $\mathcal{A}$ fails to fabricate the valid messages for $msg_{VR_1}$, $msg_{VR_2}$, $msg_{RC_1}$ and $msg_{RC_2}$. Therefore, our proposed AAKE-BIVT is resilient against MitM attacks.

*3) ESL Attack:* During the AKE phase between vehicles $OBU_i$ and $OBU_j$, a shared secret session key $SK_{OBU_j,OBU_{j'}} (= SK_{OBU_{j'},OBU_j}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{OBU_j} \cdot PR_{OBU_{j'}} \cdot P) \parallel TS_1 \parallel TS_2)$ is established. Also at the time of AKE phase between the cluster head $OBU_i$ and the associated RSU $RSU_k$, a shared secret session key is formed for secure communication as $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{OBU_i} \cdot PR_{RSU_j} \cdot G) \parallel TS_1 \parallel TS_2)$. Similarly, during the AKE phase between $RSU_k$ and $CS_m$, a shared secret session key

**Algorithm 1** The Blockchain Consensus Algorithm for Block Verification and Addition

**Input:** $n_{CS}$: Number of CSs in CSN, $GTNX_{pl}$: global transaction pool, and $Tnx_t$: transaction threshold.
**Output:** After successful verification, $Block_k$ added to blockchain.

1: **if** $(GTNX_{pl} = Tnx_t)$ **then**
2: Leader $CS_l$ is selected through the secure leader selection algorithm [33] from the peer nodes in CSN.
3: $CS_l$ constructs block $Block_k$ as shown in Fig. 5.
4: $CS_l$ initializes $VotesCT \leftarrow 0$ and $flag_{CS_j} = 0$, $\forall\{j = 1, 2, \cdots, n_{CS}, CS_l \neq CS_j\}$.
5: For each $CS_j$, $CS_l$ selects distinct random number $rn_j$ and current timestamp $TS_j$.
6: Utilizing shared secret key $SK_{CS_l,\ CS_j}$ established in Subsection IV-E, $CS_l$ encrypts voting request $VoteRQ = EC_{SK_{CS_l,\ CS_j}}(VoteRQ, rn_j)$ and computes authentication parameter $AP_{lj} = h(VoteRQ \| rn_j \| TS_j)$.
7: $CS_l$ fabricates $\{Block_k, EC_{SK_{CS_l,CS_j}}(VoteRQ, rn_j), AP_{lj}, TS_j\}$ and forwards it to all other CSs $CS_j$, $\{j = 1, 2, \cdots, n_{CS}, CS_l \neq CS_j\}$ via a public channel.
8: **for** each CS $CS_j$ **do**
9: Suppose $CS_j$ obtains message at time $TS'_j$.
10: **if** $(|\ TS_j - TS'_j\ |) < \Delta T$ **then**
11: On received $Block_k$, $CS_j$ verifies $MRT_k$, $BKHash_k$ and $SigBK_k$.
12: By utilizing the shared secret key $SK_{CS_j,CS_l}$, it decrypts the request as $(VoteRQ^*, rn_j^*) = DC_{SK_{CS_j,CS_l}}(EC_{SK_{CS_l,CS_j}}(VoteRQ, rn_j))$.
13: **if** $(h(VoteRQ^* \| rn_j^* \| TS_j) = AP_{lj})$ **then**
14: Pick timestamp $TS_j^{**}$, encrypt $(VTStatus)$ as $EC_{SK_{CS_j,CS_l}}(rn_j^*, VTStatus)$ and compute $APjl = h(rn_j^* \| VTStatus \| TS_j^{**})$. Next, forward $\{EC_{SK_{CS_j,CS_l}}(rn_j^*, VTStatus), APjl, TS_j^{**}\}$ to $CS_l$.
15: **end if**
16: **end if**
17: **end for**
18: **for** each received message from $CS_j$ **do**
19: $CS_l$ computes $(rn_j^{\#}, VTStatus_j^{\#}) = DC_{SK_{CS_l,CS_j}}(EC_{SK_{CS_j,\ CS_l}}(rn_j^*, VTStatus))$.
20: **if** $(APjl = h(rn_j^{\#} \| VTStatus_j^{\#} \| TS_j^{**}))$ **then**
21: **if** $(((rn_j^{\#} = rn_j)$ **and** $(VTStatus = valid))$ **and** $flag_{CS_j} = 0)$ **then**
22: $CS_l$ sets $VotesCT = VotesCT + 1$ **and** $flag_{CS_j} = 1$.
23: **end if**
24: **end if**
25: **end for**
26: **if** $(VotesCT > 50\%)$ **then**
27: Transaction enters to the next round.
28: **if** $(VotesCT < 80\%)$ **then**
29: **go to** Step 1.
30: **else**
31: Broadcast the committed message to all other CSs, $Block_k$ is added to the blockchain, and stop the consensus process.
32: **end if**
33: **end if**
34: **end if**

$SK_{RSU_j,CS_m}(= SK_{CS_m,RSU_j}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{RSU_j} \cdot PR_{CS_m} \cdot G) \parallel SDK_{RSU_j} \parallel SP \parallel TS_1 \parallel TS_2)$ is established for secure communication.

Consider the case of establishing the shared secret session key for secure communication between $OBU_i$ and $OBU_j$ presented in Subsection IV-D1. Under the CK-adversary model as stated in Subsection III-B, $\mathcal{A}$ may get hold of short-term secrets, such as $R_{\text{AKE1}}$, $R_{\text{AKE2}}$, $TS_1$ and $TS_2$. To construct the session key, however, $\mathcal{A}$ also requires the knowledge of long-term secrets, i.e., $PR_{OBU_i}$ and $PR_{OBU_j}$, of the involved parties. But it is a computationally infeasible task for $\mathcal{A}$ to obtain the required long-term secrets. The same is also true for the construction of SK between CH $OBU_i$ and associated RSU as well as between the RSU and its associated CS. Moreover, compromising the current SK does not reveal the past and future SK due to the randomness and distinctness in each session. Therefore, our proposed AAKE-BIVT can successfully shield both forward and backward secrecy accompanying the SK security. Hence, the AAKE-BIVT is resilient against ESL attacks.

*4) Anonymity and Untraceability Preservation:* As stated in the threat model of Subsection III-B, $\mathcal{A}$ can seize the communicated messages $msg_{VV_1}$ and $msg_{VV_2}$ during the AKE phase between vehicles $OBU_i$ and $OBU_j$ over public insecure channels. Without the knowledge of the secret parameters $PR_{OBU_i}$, $PR_{OBU_j}$, $R_{\text{AKE1}}$ and $R_{\text{AKE2}}$, however, it is a computationally challenging task for $\mathcal{A}$ to infer the identities of both $OBU_i$ and $OBU_j$ in polynomial time. Therefore, this ensures that our proposed AAKE-BIVT can preserve the anonymity feature. Regarding the untraceability feature, it is worth noting that the nature of the communicated messages is dynamic, which are calculated by utilizing current timestamps and random nonces. Moreover, by utilizing the collision-resistant cryptographic hash function $h(\cdot)$, it is impractical for $\mathcal{A}$ to trace the transmitted messages. Thus, the AAKE-BIVT also preserves the untraceability feature.

Similarly, the AKE schemes between CH and its associated RSU as well as between RSU and its associated CS also ensure the anonymity and untraceability properties.

*5) Stolen Smart Card Attack:* If adversary $\mathcal{A}$ obtains a stolen or lost smart card $SC$, it can extract the parameters $\{A_i, C_i, RP_{OBU_i}, Q_i, W_i\}$ stored in $SC$'s memory by performing PA attacks. From the extracted information, however, $\mathcal{A}$ cannot obtain the secret credentials, such as $ID_{OBU_i}$, $PR_{OBU_i}$ and $RN_i$. To procure these secret parameters, $\mathcal{A}$ requires computing $R_i = PUF(C_i)$, $SDK_{OBU_i} = Rep(R_i, RP_{OBU_i})$, $ID_{OBU_i} = h(SDK_{OBU_i})$, $RN_i = A_i \oplus h(PW^l_{VO_i} \parallel ID_{OBU_i})$, $RPW^l_{VO_i} = h(PW^l_{VO_i} \parallel RN_i)$ and $PR_{OBU_i} = Q_i \oplus h(RPW_{VO_i} \parallel ID_{OBU_i})$. But to perform these computations, $\mathcal{A}$ requires the secret parameter $PW_{VO_i}$, which is known only to $VO_i$. Thus, our AAKE-BIVT is resilient against stolen smart card attacks.

*6) Impersonation Attacks:* Assume that adversary $\mathcal{A}$ acts as a licit CH (for instance, $OBU_i$) to its associated RSU. In such a situation, $\mathcal{A}$ may then attempt to compose a legitimate AKE request message $msg_{VR_1} = \{M_1, Auth_1, BSC, TS_1\}$ to impersonate $OBU_i$. $\mathcal{A}$ may initially pick a timestamp $TS^*_1$ and a random secret $R^*_{\text{AKE1}}$ to accomplish this purpose. Next, $\mathcal{A}$ may try to form $M_1 = (R^*_{\text{AKE1}} \cdot P) \oplus PB_{OBU_i}$, $BSC = R^*_{\text{AKE1}} \cdot PB_{RSU_j}$ and $Auth_1 = h((R^*_{\text{AKE1}} \cdot P) \parallel BSC \parallel TS^*_1)$. However, in order to generate a legitimate message, $\mathcal{A}$ requires the knowledge of the random secret $R_{\text{AKE1}}$ of $OBU_i$. Therefore, $\mathcal{A}$ cannot impersonate $OBU_i$. Similarly, $\mathcal{A}$ is incapable of impersonating CS and RSU. Thus, the proposed AAKE-BIVT can withstand impersonation attacks.

*7) DoS Attack:* Vehicle owner $VO_i$ puts smart card $SC$ into $OBU_i$ and enters the password into smart application $SA_i$ during the vehicle user login phase. Then, $SA_i$ calculates $R_i = PUF(C_i)$, $SDK_{OBU_i} = Rep(R_i, RP_{OBU_i})$, $ID_{OBU_i} = h(SDK_{OBU_i})$, $RN_i = A_i \oplus h(PW^l_{VO_i} \parallel ID_{OBU_i})$, $RPW^l_{VO_i} = h(PW^l_{VO_i} \parallel RN_i)$, $PR_{OBU_i} = Q_i \oplus h(RPW_{VO_i} \parallel ID_{OBU_i})$ and $W'_i = h(RPW^l_i \parallel PR_{OBU_i})$ and then check if $W'_i \stackrel{?}{=} W_i$ holds. If so, $VO_i$ is successfully signed in and can now carry out future communications. In this configuration, the login process just uses the $OBU_i$ side and does not utilize the bandwidth of the other party (such as the other vehicle and RSU). Hence, our AAKE-BIVT eliminates DoS attacks.

*8) Non-Linkability:* The proposed AAKE-BIVT can guarantee non-linkability for multiple messages from the same source. For instance, in the AKE phase between vehicles $OBU_j$ and $OBU_{j'}$, the messages exchanged are inherently dynamic because random numbers and timestamps are used; thus, there is no correlation between different interaction information of the same vehicle. So the adversary cannot extract sensitive credentials from different information of the same vehicle. Thus, the AKE scheme between vehicles preserves the non-linkability feature. Similarly, the AKE schemes between the CH and its associated RSU as well as between the RSU and its associated CS also ensure the non-linkability property.

*9) Data Tempering Attack at CSN:* CS $CS_k$ obtains data from the associated RSU $RSU_k$, encrypts it, and temporarily stores it in a $GTNX_{pl}$. When $GTNX_{pl}$ reaches a threshold, a leader is selected from the CSN, which constructs a block and broadcasts it into the CSN for consensus. Additionally, the block is included in the blockchain once consensus has been attained. Since blockchain cannot be altered, $\mathcal{A}$ cannot change the block data. Thus, the AAKE-BIVT is resilient against data tempering attacks at CSN.

### *B. Formal Security Analysis Using ROR Model*

For AKE schemes, the ROR model is prominent formal security analyzing mechanism and is regarded as a robust SK security validation technique. Theorem 1 demonstrates the SK security for the devised AAKE-BIVT. Suppose that $\Psi^d_{VO}$, $\Psi^e_{OBU}$, $\Psi^f_{RSU}$ and $\Psi^g_{CS}$ signify the instances $d$, $e$, $f$ and $g$ of the participants VO, OBU, RSU and CS, respectively, alias as the oracles. In Table III, various queries are tabulated, which adversary $\mathcal{A}$ can perform.

*Theorem 1: $\mathcal{A}$ is launching an attack against the AAKE-BIVT in polynomial time ($t_{poly}$). Let $HQ$, $|Hash|$, $Adv^{ECDDHP}_{\mathcal{A}}(t_{poly})$, $Q_s$ and $|Dic|$ represent $Hash$ queries, $Hash$ output range, the advantage of $\mathcal{A}$ in breaching the Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP), Send queries and the password dictionary, respectively. Then,*

TABLE III
DESCRIPTION OF VARIOUS QUERIES

| Query | Description |
|---|---|
| $Send(\Psi, msg)$ | This query enables $\mathcal{A}$ to execute an active attack by transmitting a message $msg$ to $\Psi$ and acquiring a response accordingly. |
| $Execute(\Psi^e_{OBU}, \Psi^f_{RSU})$ | This query enables $\mathcal{A}$ to eavesdrop on all the transmitted messages between $\Psi^e_{OBU}$ and $\Psi^f_{RSU}$. |
| $Hash(\Psi, msg)$ | With this query, $\mathcal{A}$ can dispatch a hashed message $msg$ to $\Psi$ and acquire a response accordingly. |
| $CorruptSC(\Psi^d_{VO})$ | By executing this query, $\mathcal{A}$ can procure the stored secret parameters in lost/stolen $SC$. |
| $CorruptOBU(\Psi^e_{OBU})$ | By utilizing this query, $\mathcal{A}$ can access the secret information stored in lost or compromised $OBU$. |
| $Reveal(\Psi)$ | By exercising this query, $\mathcal{A}$ can reveal the secret SK, established between $\Psi$ and its partner. |
| $Test(\Psi)$ | By employing this query, $\mathcal{A}$ strives to guess an SK by dispatching a request to $\Psi$. The response of $\Psi$ is probabilistic like toss coin $C$. |

*approximated advantage of $\mathcal{A}$ in breaching the security of the AAKE-BIVT for procuring the SK between the communicating participants, denoted as $Adv^{AAKE-BIVT}_{\mathcal{A}}(t_{poly})$, is upper bounded by the following inequality*

$$Adv^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) \leq \frac{HQ^2}{|Hash|} + \frac{2 \cdot Q_s}{|Dic|} + 2 \cdot Adv^{ECDDHP}_{\mathcal{A}}(t_{poly}). \quad (1)$$

*Proof:* Let $Game_k$, for $k = \{0, 1, 2, 3, 4\}$, signifies a series of games played by $\mathcal{A}$ to breach the security of the AAKE-BIVT and $Succ_k$ indicates the probability of success in which $\mathcal{A}$ wins the game $Game_k$ in time $t_{poly}$. Specifically, we have

$Game_0$: This game simulates the real attack by $\mathcal{A}$ against the devised AAKE-BIVT. In this game, the judgment is obtained by flipping an unbiased coin, and hence we have

$$\text{Adv}^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) = |2 \cdot \text{Prob}[Succ_0] - 1|. \quad (2)$$

$Game_1$: In this game, $\mathcal{A}$ performs eavesdropping against the AAKE-BIVT with the help of $Execute$ query. In the end, $\mathcal{A}$ runs the $Test$ query. Further, $\mathcal{A}$ verifies the output is a valid SK or a random output. Note that the SK between the CH $OBU_i$ and $RSU_j$ is computed as $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{OBU_i} \cdot PR_{RSU_j} \cdot G) \parallel TS_1 \parallel TS_2)$, which includes both the short-term secrets and long-term secrets. It is a computationally challenging task for $\mathcal{A}$ to procure the SK, as these secret parameters are concealed in the transmitted messages $msg_{VR_1}$ and $msg_{VR_2}$. Thus, eavesdropping on these messages does not benefit the task of stealing the session key $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i})$, and the winning probability of $Game_1$ is not increased from that of $Game_0$, namely,

$$\text{Prob}[Succ_1] = \text{Prob}[Succ_0]. \quad (3)$$

$Game_2$: In this game, $\mathcal{A}$ performs an active attack by executing the $Hash$ and $Send$ queries. $\mathcal{A}$ can check for hash collisions by running multiple $Hash$ queries in order to accomplish this. There is no collision when $\mathcal{A}$ runs the $Send$ query because each exchanged message in the devised scheme contains timestamps and random numbers; therefore, the collision probability in hash outputs is nearly zero. Thus, the birthday paradox exhibits outcome as follows

$$|\text{Prob}[Succ_2] - \text{Prob}[Succ_1]| \leq \frac{HQ^2}{2|Hash|}. \quad (4)$$

$Game_3$: In this game, $\mathcal{A}$ executes an active attack and attempts to procure the session key $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i})$ using all the eavesdropped messages $msg_{VR_1}$ and $msg_{VR_2}$ between $OBU_i$ and $RSU_j$ as well as the other secret parameters acquired from the games mentioned earlier. In order to do this, $\mathcal{A}$ must compute $SK_{OBU_i,RSU_j} (= SK_{RSU_j,OBU_i}) = h((R_{\text{AKE1}} \cdot P) \parallel R_{\text{AKE2}} \parallel (PR_{OBU_i} \cdot PR_{RSU_j} \cdot G) \parallel TS_1 \parallel TS_2)$. In other words, $\mathcal{A}$ must solve the ECDDHP for acquiring the SK. It follows that

$$|\text{Prob}[Succ_3] - \text{Prob}[Succ_2]| \leq \text{Adv}^{ECDDHP}_{\mathcal{A}}(t_{poly}). \quad (5)$$

$Game_4$: This game mimics lost/stolen $SC$ attacks, insider attacks, and password guessing attacks. $\mathcal{A}$ obtains $\{A_j, C_j, RP_{OBU_j}, Q_j, W_j\}$ from a lost/stolen $SC$ and $\{A_j, C_j, RP_{OBU_j}, Q_j, W_j\}$ from $OBU_i$'s memory by utilizing $CorruptSC$ and $CorruptOBU$ queries, respectively, to reveal $VO_i$'s credential $\{PW_{VO_i}\}$ or produce an access request. To succeed in this game, $\mathcal{A}$ requires to guess $PW_{VO_i}$ with a bounded number of guesses from $Dic$, and hence

$$|\text{Prob}[Succ_4] - \text{Prob}[Succ_3]| \leq \frac{Q_s}{|Dic|}. \quad (6)$$

As $\mathcal{A}$ has performed all the games, it executes a $Test$ query. Further, a fair coin is flipped to deduce the semantic security of SK, and hence

$$\text{Prob}[Succ_4] = \frac{1}{2}. \quad (7)$$

Therefore, from (2) we have

$$\frac{1}{2}\text{Adv}^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) = \left|\text{Prob}[Succ_0] - \frac{1}{2}\right|. \quad (8)$$

Using (7) and (8) as well as noting (3), we obtain

$$\begin{aligned}\frac{1}{2}\text{Adv}^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) &= |\text{Prob}[Succ_0] - \text{Prob}[Succ_4]| \\ &= |\text{Prob}[Succ_1] - \text{Prob}[Succ_4]|.\end{aligned} \quad (9)$$

By applying the famous triangular inequality to (9), we have

$$\begin{aligned}\frac{1}{2}\text{Adv}^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) \leq\ & |\text{Prob}[Succ_1] - \text{Prob}[Succ_2]| \\ &+ |\text{Prob}[Succ_2] - \text{Prob}[Succ_3]| \\ &+ |\text{Prob}[Succ_3] - \text{Prob}[Succ_4]|.\end{aligned} \quad (10)$$

Substituting (4), (5) and (6) into (10) leads to

$$\frac{1}{2}\text{Adv}^{AAKE-BIVT}_{\mathcal{A}}(t_{poly}) \leq \frac{HQ^2}{2 \cdot |Hash|} + \frac{Q_s}{|Dic|} + \text{Adv}^{ECDDHP}_{\mathcal{A}}(t_{poly}), \quad (11)$$

that is, (1). This completes the proof. ■

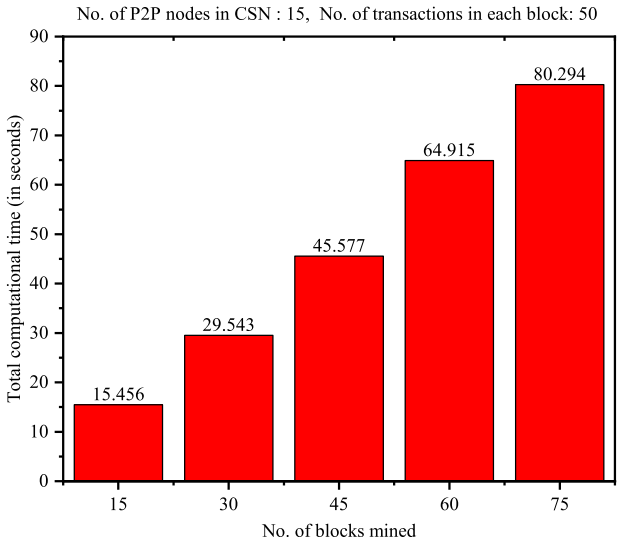


Fig. 6. Blockchain simulation results for Case 1.

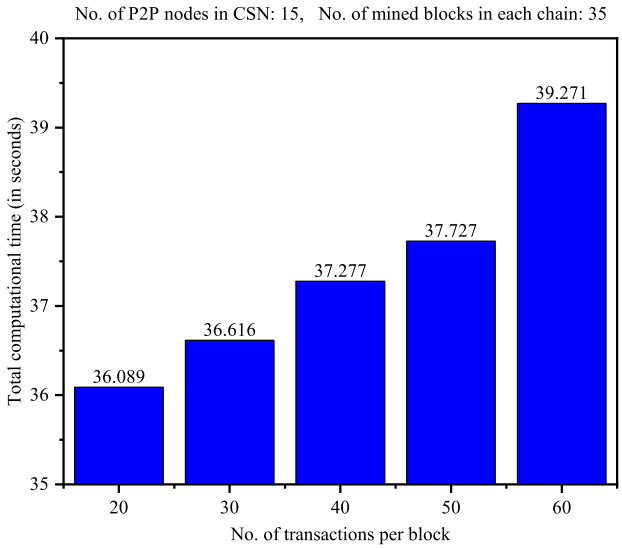


Fig. 7. Blockchain simulation results for Case 2.

## VI. Blockchain Implementation

In this section, we furnish the blockchain implementation of the devised AAKE-BIVT. The experiments were performed over a platform having Intel® Core ™ i7-6700 CPU@3.4GHz; RAM@8 GiB; OS@Ubuntu 20.04.2 LTS, and for script implementation, Node.js® framework in Visual Studio (VS) Code (version 1.60) integrated development environment (IDE) were utilized [36], [37].

In the simulation, the sizes of the block version, timestamp, previous block hash, Merkle tree root, proposer identity, the public key of the proposer, block payload, current block hash (SHA-256) and ECDSA signature are 32, 32, 256, 256, 160, 320, $640 \cdot n_t$, 256, and 320 bits, respectively. Therefore, the total size of the block becomes $1632 + 640 \cdot n_t$, where $n_t$ is the total number of transactions stored in a block. Two cases are considered.

**Case 1**: In this case, we evaluate the performance by considering the total number of P2P nodes in the CSN is 15 and the number of transactions in each block is 50. The simulation results depicted in Fig. 6 illustrate the computational time (s) as the function of the number of blocks mined. As expected, the computational time increases as the number of blocks mined increases.

**Case 2**: In this case, we evaluate the performance by considering the number of blocks mined in each chain is 35 and the total number of P2P nodes in the CSN is 15. The simulation results furnished in Fig. 7 demonstrate that the total computational time (s) increases as the number of transactions stored in a block for fixed chain length varies.

## VII. Comparative Analysis

This section conducts rigorous comparative analysis on the communication and computation overheads during the AKE phase as well as the security and functionality features among the devised AAKE-BIVT and the existing schemes of Liu et al. [11], Tan and Chung [19], Li et al. [21], Vasudev et al. [22], Vangala et al. [23], and Chattaraj et al. [24]. As the enrollment and password updation phases are infrequent, the overheads required in these phases are not included.

TABLE IV
Cryptography Operations Utilized for Analysis

| Cryptographic operation | Symbol | Raspberry PI-3 | Server |
|---|---|---|---|
| Bilinear pairing | $T_{bpo}$ | 32.084 ms | 4.716 ms |
| ECC point addition | $T_{eca}$ | 0.016 ms | 0.002 ms |
| ECC point multiplication | $T_{ecm}$ | 2.288 ms | 0.674 ms |
| Fuzzy extractor function | $T_{fe} \approx T_{ecm}$ | 2.288 ms | 0.674 ms |
| Map to elliptic curve point | $T_{mtp}$ | 0.385 ms | 0.114 ms |
| Modular addition | $T_{ma}$ | 0.010 ms | 0.001 ms |
| Modular exponentiation | $T_{exp}$ | 0.228 ms | 0.039 ms |
| Modular multiplication | $T_{mul}$ | 0.011 ms | 0.002 ms |
| Physical unclonable function | $T_{puf}$ | 0.4 $\mu$s | - |
| SHA-256 hash function | $T_h$ | 0.309 ms | 0.055 ms |
| Symmetric decryption | $T_{sd}$ | 0.014 ms | 0.003 ms |
| Symmetric encryption | $T_{se}$ | 0.018 ms | 0.003 ms |

### A. Computation Overheads Comparison

We utilize the existing experimental execution times of various cryptographic operations on distinct platforms reported in [24], and for PUF, we use the results produced in [35]. Table IV lists various cryptographic operations, their symbols, and execution times on different platforms. Note that the execution time for XOR operation is negligible, and we do not consider it in the computational overheads.

We first analyze the computational overheads of our AAKE-BIVT. In V2CH scenario, the computational overheads imposed by vehicle and associated CH are $3T_h + 3T_{ecm}$ and $2T_h + 3T_{ecm}$, respectively. The total computational overhead for V2CH scenario is therefore $6T_h + 5T_{ecm} \approx 13.294$ ms. For CH2RSU scenario, CH and associated RSU impose the

TABLE V
COMPUTATION OVERHEAD OF PROPOSED AND RELATED SCHEMES

| Scheme | Based on | OBU/Vehicle/CH | RSU/Server/CS |
|---|---|---|---|
| Liu *et al.* [11] | V2RSU | $6T_h + 3T_{mul} + 7T_{ecm} + 2T_{eca} \approx 17.935$ ms | $4T_h + 2T_{mul} + 4T_{ecm} + 3T_{eca} + T_{bpo} \approx 7.642$ ms |
| Tan and Chung [19] | V2RSU | $n(8T_h + 10T_{ecm} + 2T_{eca}) \approx 5076.80$ ms | $n(6T_h + 7T_{ecm} + 2T_{exp} + 4T_{bpo}) \approx 4798$ ms |
| | V2V | $m(3T_h + 2T_{ecm}) \approx 550.03$ ms | $(m+2)T_h + (3m+1)T_{ecm} \approx 208.484$ ms |
| Li *et al.* [21] | V2RSU | $7T_h + 10T_{ecm} + 4T_{eca} \approx 25.107$ ms | $7T_h + 10T_{ecm} + 4T_{eca} \approx 7.133$ ms |
| Vasudev *et al.* [22] | V2V | $6T_h + T_{se} + T_{sd} \approx 1.886$ ms | — |
| | V2RSU | $3T_h + T_{sd} \approx 0.941$ ms | $3T_h + T_{se} \approx 0.168$ ms |
| Vangala *et al.* [23] | V2CH | $2(5T_h + 2T_{eca} + 6T_{ecm}) \approx 30.61$ ms | — |
| | CH2RSU | $7T_h + 2T_{eca} + 6T_{ecm} \approx 15.923$ ms | $8T_h + 2T_{eca} + 6T_{ecm} \approx 4.488$ ms |
| Chattaraj *et al.* [24] | V2RSU | $5T_h + 5T_{ecm} + T_{eca} \approx 13.001$ ms | $3T_h + 5T_{ecm} + T_{eca} \approx 3.537$ ms |
| | V2CH | $2(T_{eca} + 4T_h + 5T_{ecm}) \approx 25.384$ ms | — |
| | RSU2CS | — | $2(T_{poly} + 6T_h) \approx 6.66$ ms |
| Proposed AAKE-BIVT | V2CH | $6T_h + 5T_{ecm} \approx 13.294$ ms | — |
| | CH2RSU | $3T_h + 3T_{ecm} \approx 7.791$ ms | $5T_h + 2T_{ecm} + T_{puf} + T_{fe} \approx 2.297$ ms |
| | RSU2CS | — | $13T_h + 5T_{ecm} + 2T_{fe} + T_{puf} + T_{sd} \approx 5.436$ ms |

**Note:** The number of neighboring vehicles and the number of vehicles ready to be in a group are represented by $n$ and $m$, respectively, in the scheme of Tan and Chung [19]; $t$-degree polynomial requires $t$ modular additions and $t$ modular multiplications, i.e., $T_{poly} = tT_{mul} + tT_{ma}$, in the scheme of Chattaraj *et al.* [24]. Furthermore, $m = 100$, $n = 200$, and $t = 1000$.

computational overheads of $3T_h + 3T_{ecm} \approx 7.791$ ms and $5T_h + 2T_{ecm} + T_{puf} + T_{fe} \approx 2.297$ ms, respectively. Thus, the cumulative computational overhead of CH2RSU scenario is $8T_h + 5T_{ecm} + T_{puf} + T_{fe} \approx 10.088$ ms. Lastly, for RSU2CS scenario, RSU and associated CS impose the computational overheads of $7T_h + 3T_{ecm} + T_{puf} + T_{fe}$ and $6T_h + 2T_{ecm} + T_{fe} + T_{sd}$, respectively, and thus the cumulative overhead is $13T_h + 5T_{ecm} + 2T_{fe} + T_{puf} + T_{sd} \approx 5.436$ ms.

The computation overheads of our proposed AAKE-BIVT are compared with those of the six existing schemes in Table V. It can be seen that, with the exception of the scheme of Vasudev et al. [22], our proposed AAKE-BIVT requires far fewer computational overheads as compared to the other five existing schemes. Although our scheme necessitates more computational overhead as compared to the scheme of [22], it offers more functionality and security features over the scheme of [22] (see Table VII).

### B. Communication Overheads Comparison

To measure the communication overheads of our AAKE-BIVT in the three scenarios, we contemplate the number of messages exchanged and bits transmitted over the communication channel between the communicating entities. We make a reasonable assumption that the bit-lengths of the real identity, random number, elliptic curve point, timestamp and hash function are 128, 128, 320, 32 and 256 bits, respectively. In V2CH scenario, the communication overheads for two messages $msg_{VV_1} = \{M_1, Auth_1, BSC, TS_1\}$ and $msg_{VV_2} = \{M_2, Auth_3, TS_2\}$ demand $(320 + 256 + 320 + 32) = 928$ bits and $(320 + 256 + 32) = 608$ bits, respectively, which add to a total of 1536 bits. Again in CH2RSU scenario, two messages $msg_{VR_1} = \{M_1, Auth_1, BSC, TS_1\}$ and $msg_{VR_2} = \{M_2, Auth_3, TS_2\}$ demand $(320+256+320+32) = 928$ bits and $(320 + 256 + 32) = 608$ bits, respectively, which add to a total of 1536 bits. In RSU2CS scenario, two messages $msg_{RC_1} = \{Z_3, BSC, Z_4, Z_5, TS_1\}$ and $msg_{RC_2} = \{Z_6, Z_7, TS_2\}$ demand $(320+320+256+256+32) = 1184$ bits and $(320 + 256 + 32) = 608$ bits, respectively, which add to a total of 1792 bits.

TABLE VI
COMMUNICATION OVERHEAD OF PROPOSED AND RELATED SCHEMES

| Scheme | Based on | Number of messages | Total overhead (bits) |
|---|---|---|---|
| Liu *et al.* [11] | V2RSU | 3 | 2752 |
| Tan and Chung [19] | V2RSU | $2n + 1$ | $992 + 1344n$ |
| | V2V | $3m + 2$ | $3584m$ |
| Li *et al.* [21] | V2RSU | 5 | 5536 |
| Vasudev *et al.* [22] | V2V | 2 | 1024 |
| | V2RSU | 2 | 1280 |
| Vangala *et al.* [23] | V2CH | 2 | 1856 |
| | CH2RSU | 3 | 2400 |
| Chattaraj *et al.* [24] | CH2RSU | 3 | 2560 |
| | V2CH | 3 | 2464 |
| | RSU2CS | 3 | 1376 |
| Proposed AAKE-BIVT | V2CH | 2 | 1536 |
| | CH2RSU | 2 | 1536 |
| | RSU2CS | 2 | 1792 |

The communication overheads of AAKE-BIVT and six existing schemes are compared in Table VI, which demonstrates that, with the exception of the scheme [22], [24], the devised AAKE-BIVT imposes lower communication overhead than the other four schemes, although our scheme necessitates higher communication overhead in V2CH or V2V scenario than the scheme of [22]. Moreover, the apparent excess communication overhead vis-a-vis [24] is amongst RSU2CS. The slightly higher communication overhead is well justified because our proposed AAKE-BIVT sustains more security and functionality features (see Table VII).

### C. Functionality and Security Features Comparison

In Table VII, our AAKE-BIVT is compared with the existing schemes of Liu et al. [11], Tan and Chung [19],

TABLE VII
SECURITY AND FUNCTIONALITY FEATURES COMPARISON

| Scheme | $\mathcal{FE}_1$ | $\mathcal{FE}_2$ | $\mathcal{FE}_3$ | $\mathcal{FE}_4$ | $\mathcal{FE}_5$ | $\mathcal{FE}_6$ | $\mathcal{FE}_7$ | $\mathcal{FE}_8$ | $\mathcal{FE}_9$ | $\mathcal{FE}_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Liu *et al.* [11] | × | × | ✓ | ✓ | × | ✓ | ✓ | × | ✓ | × |
| Tan and Chung [19] | ✓ | × | ✓ | ✓ | × | ✓ | ✓ | × | ✓ | ✓ |
| Li *et al.* [21] | × | × | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × |
| Vasudev *et al.* [22] | × | × | ✓ | × | × | ✓ | ✓ | ✓ | ✓ | × |
| Vangala *et al.* [23] | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |
| Chattaraj *et al.* [24] | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |
| Our AAKE-BIVT | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

✓: support the feature or the scheme is secure; × : does not support the feature or the scheme is insecure.

Li et al. [21], Vasudev et al. [22], Vangala et al. [23], and Chattaraj et al. [24] based on the set of ten functionality and security features, namely, $\mathcal{FE}_1$: support blockchain solution; $\mathcal{FE}_2$: dynamic node addition phase; $\mathcal{FE}_3$: replay attack; $\mathcal{FE}_4$: MitM attack; $\mathcal{FE}_5$: ESL attack; $\mathcal{FE}_6$: anonymity and untraceability preservation; $\mathcal{FE}_7$: stolen smart card attack; $\mathcal{FE}_8$: impersonation attacks; $\mathcal{FE}_9$: DoS attack; and $\mathcal{FE}_{10}$: data tempering attack. It is worth mentioning that our devised AAKE-BIVT and the schemes of Tan and Chung [19], Vangala et al. [23] and Chattaraj et al. [24] assist blockchain solution. It is clear that our proposed AAKE-BIVT offers more functionality and security features than the other existing schemes.

### D. Critical Discussion

In the IoVs environment, various entities communicate via insecure or open channels, which are exposed to various security assaults and threats. In order to address these security concerns, we designed AAKE-BIVT, which simultaneously permits the AKE scheme among V2CH, CH2RSU, and RSU2CS. These AKE schemes enable vehicles, RSUs, and CSs to authenticate and establish a session key to secure communication. Rigorous security analysis demonstrates that our proposed AAKE-BIVT is resilient against potential security attacks and satisfies session-key security. The PUF trait enables smart vehicles and RSUs to prevent tampering from physical attacks. The comparative analysis presented in the previous subsection shows that our proposed AAKE-BIVT reasonably improves performance by minimizing the computational and communication overheads and by adding additional security and functional features as compared to most benchmarks. The reason for the improved performance of AAKE-BIVT is that the design goals of our proposed schemes are to reduce the communication and computational overheads of AKE procedures, for which we use ultra-lightweight cryptography technology composed of XOR, ECC, hash function, and symmetric encryption/decryption accompanying PUF.

Nevertheless, the credential revocation mechanism to achieve conditional privacy protection is not considered in our proposed AAKE-BIVT. Therefore, for future work, we plan to design a blockchain-based authentication scheme with conditional privacy that supports the revocation mechanism.

## VIII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we have proposed an anonymous authenticated key exchange scheme for blockchain-enabled IoVs applications in smart transportation, called AAKE-BIVT. Our proposed AAKE-BIVT establishes authenticated key exchange between vehicles, between cluster head and its associated RSUs, between RSUs and cloud servers (CSs), as well as offers the key management among CSs. The data from the vehicles are securely routed to a cluster head, which in turn routes the data to a nearby RSU. CS then collects data from its associated RSUs and creates transactions in a secure environment. Additionally, transactions are aggregated into blocks by CS in a peer-to-peer CS network. The blocks are then mined, verified, and incorporated into the blockchain network.

An extensive security analysis that includes informal analysis and formal analysis through the random oracle model has confirmed that the proposed AAKE-BIVT is resistant to a wide range of potential security attacks commonly occurring in IoV environments. An in-depth comparative analysis has further demonstrated that our AAKE-BIVT outperforms many existing leading-edge schemes, in terms of computation and communication overheads, as well as offers more functionality and security features. As a future work, we intend to design a blockchain-based handover authentication scheme that reduces the overhead caused by the reauthentication of vehicles. Furthermore, since the inception of the blockchain-enabled IoVs, new application scenarios are emerging, such as data protection and management, content broadcasting, vehicle management, and traffic control and management. Our future work will also include the design of a blockchain-based authentication scheme with conditional privacy to support revocation mechanism.

## REFERENCES

[1] Z. A. Hamid, U. H. Zamzuri, and D. K. Limbu, "Internet of Vehicle (IoV) applications in expediting the implementation of smart highway of autonomous vehicle: A survey," in *Performability Internet Things*, F. Al-Turjman, Ed. Cham, Switzerland: Springer, 2019, pp. 137–157.

[2] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—Architectures, enabling technologies, applications, and development areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018.

[3] S. Ullah, G. Abbas, Z. H. Abbas, M. Waqas, and M. Ahmed, "RBO-EM: Reduced broadcast overhead scheme for emergency message dissemination in VANETs," *IEEE Access*, vol. 8, pp. 175205–175219, 2020.

[4] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: A survey and future perspectives," *IEEE Commun. Mag.*, vol. 54, no. 2, pp. 98–104, Feb. 2016.

[5] S. Ullah, G. Abbas, M. Waqas, Z. H. Abbas, S. Tu, and I. A. Hameed, "EEMDS: An effective emergency message dissemination scheme for urban VANETs," *Sensors*, vol. 21, no. 5, pp. 1–19, 2021.

[6] Allied Market Research. *Internet of Vehicle Market Outlook: 2024*. Accessed: Dec. 20, 2021. [Online]. Available: https://www.alliedmarketresearch.com/internet-of-vehicles-market

[7] M. Wazid, B. Bera, A. K. Das, S. P. Mohanty, and M. Jo, "Fortifying smart transportation security through public blockchain," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16532–16545, Sep. 2022.

[8] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, and L. H. Koh, "Blockchain for the Internet of Vehicles towards intelligent transportation systems: A survey," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4157–4185, Mar. 2021, doi: 10.1109/JIOT.2020.3028368.

[9] A. Badshah, M. Waqas, F. Muhammad, G. Abbas, and Z. H. Abbas, "A novel framework for smart systems using blockchain-enabled Internet of Things," *IT Prof.*, vol. 24, no. 3, pp. 73–80, May 2022.

[10] P. Bagga, A. K. Das, S. Member, and Y. Park, "Authentication protocols in Internet of Vehicles: Taxonomy, analysis, and challenges," *IEEE Access*, vol. 8, pp. 54314–54344, 2020.
[11] J. Liu, Q. Li, R. Sun, X. Du, and M. Guizani, "An efficient anonymous authentication scheme for Internet of Vehicles," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6.
[12] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2467–2476, Feb. 2017.
[13] P. Vijayakumar, M. S. Obaidat, M. Azees, S. H. Islam, and N. Kumar, "Efficient and secure anonymous authentication with location privacy for IoT-based WBANs," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2603–2611, Apr. 2020.
[14] A. Maria, V. Pandi, J. D. Lazarus, M. Karuppiah, and M. S. Christo, "BBAAS: Blockchain-based anonymous authentication scheme for providing secure communication in VANETs," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, Feb. 2021.
[15] L. Wei, J. Cui, H. Zhong, Y. Xu, and L. Liu, "Proven secure tree-based authenticated key agreement for securing V2V and V2I communications in VANETs," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3280–3297, Sep. 2022.
[16] L. Wei, J. Cui, H. Zhong, I. Bolodurina, and L. Liu, "A lightweight and conditional privacy-preserving authenticated key agreement scheme with multi-TA model for fog-based VANETs," *IEEE Trans. Dependable Secure Comput.*, early access, Dec. 14, 2021, doi: 10.1109/TDSC.2021.3135016.
[17] R. Vinoth, L. J. Deborah, P. Vijayakumar, and N. Kumar, "Secure multifactor authenticated key agreement scheme for industrial IoT," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3801–3811, Mar. 2021.
[18] X. Xia, S. Ji, P. Vijayakumar, J. Shen, and J. J. P. C. Rodrigues, "An efficient anonymous authentication and key agreement scheme with privacy-preserving for smart cities," *Int. J. Distrib. Sens. Netw.*, vol. 17, no. 6, Jun. 2021, Art. no. 15501477211026804.
[19] H. Tan and I. Chung, "Secure authentication and key management with blockchain in VANETs," *IEEE Access*, vol. 8, pp. 2482–2498, 2020.
[20] M. F. Moghadam, M. Nikooghadam, M. A. B. A. Jabban, M. Alishahi, L. Mortazavi, and A. Mohajerzadeh, "An efficient authentication and key agreement scheme based on ECDH for wireless sensor network," *IEEE Access*, vol. 8, pp. 73182–73192, 2020.
[21] X. Li, Y. Han, J. Gao, and J. Niu, "Secure hierarchical authentication protocol in VANET," *IET Inf. Secur.*, vol. 14, no. 1, pp. 99–110, Jan. 2020.
[22] H. Vasudev, D. Das, and A. V. Vasilakos, "Secure message propagation protocols for IoVs communication components," *Comput. Electr. Eng.*, vol. 82, Mar. 2020, Art. no. 106555.
[23] A. Vangala, B. Bera, S. Saha, A. K. Das, N. Kumar, and Y. Park, "Blockchain-enabled certificate-based authentication for vehicle accident detection and notification in intelligent transportation systems," *IEEE Sensors J.*, vol. 21, no. 14, pp. 15824–15838, Jul. 2021.
[24] D. Chattaraj, B. Bera, A. K. Das, S. Saha, P. Lorenz, and Y. Park, "Block-CLAP: Blockchain-assisted certificateless key agreement protocol for Internet of Vehicles in smart transportation," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8092–8107, Aug. 2021.
[25] M. S. Kakkasageri and S. S. Manvi, "Multiagent driven dynamic clustering of vehicles in VANETs," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 1771–1780, Nov. 2012.
[26] X. L. Wang, P. Zeng, N. Patterson, F. Jiang, and R. Doss, "An improved authentication scheme for Internet of Vehicles based on blockchain technology," *IEEE Access*, vol. 7, pp. 45061–45072, 2019.
[27] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 2, pp. 198–208, Mar. 1983.
[28] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.
[29] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Proc. EUROCRYPT*, Amsterdam, The Netherlands, Apr. 2002, pp. 337–351.
[30] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Ruhrmair, "The bistable ring PUF: A new architecture for strong physical unclonable functions," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust*, San Diego, CA, USA, Jun. 2011, pp. 134–141.
[31] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M. D. M. Yu, "Efficient fuzzy extraction of PUF-induced secrets: Theory and applications," in *Proc. CHES*, Santa Barbara, CA, USA, Aug. 2016, pp. 412–431.
[32] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," *Inf. Comput.*, vol. 146, no. 1, pp. 1–23, 1998.
[33] B. Bera, S. Saha, A. K. Das, and A. V. Vasilakos, "Designing blockchain-based access control protocol in IoT-enabled smart-grid system," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5744–5761, Apr. 2021.
[34] D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," Ripple Labs, San Francisco, CA, USA, White Paper, 2014, pp. 1–8.
[35] T. Alladi, N. Naren, G. Bansal, V. Chamola, and M. Guizani, "SecAuthUAV: A novel authentication scheme for UAV-ground station and UAV-UAV communication," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15068–15077, Dec. 2020.
[36] A. Badshah et al., "LAKE-BSG: Lightweight authenticated key exchange scheme for blockchain-enabled smart grids," *Sustain. Energy Technol. Assessments*, vol. 52, May 2022, Art. no. 102248.
[37] K. Kashish. *Implementing PBFT in Blockchain*. Accessed: Dec. 24, 2021. [Online]. Available: https://medium.com/coinmonks/implementing-pbft-in-blockchain-12368c6c9548