Efficient Blockchain-based Group Key Distribution for Secure Authentication in VANETs

Mahmoud A. Shawky, Abdul Jabbar, *Student Member, IEEE*, Muhammad Usman, Muhammad Imran, Qammer H. Abbasi, Shuja Ansari, *Senior Member, IEEE*, and Ahmad Taha, *Member, IEEE*

***Abstract*—This paper proposes a group key distribution scheme using smart contract-based blockchain technology. The smart contract's functions allow for securely distributing the group session key, following the initial legitimacy detection using public key infrastructure-based authentication. For message authentication, we propose a lightweight symmetric key cryptography-based group signature method, supporting the security and privacy requirements of vehicular ad hoc networks (VANETs). Our discussion examined the scheme's robustness against typical adversarial attacks. To evaluate the gas costs associated with smart contract's functions, we implemented it on the Ethereum main network. Finally, comprehensive analyses of computation and communication costs demonstrate the scheme's effectiveness.**

***Index Terms*—Authentication, Blockchain technology, Group key distribution, Public key infrastructure, Smart contracts.**

I. Introduction

Intelligent transportation systems are highly beneficial in improving transportation safety and increasing productivity by offering direct communication from vehicle to vehicle (V2V) and from vehicle to infrastructure (V2I) [1]. In this context, each vehicle in the vehicular ad hoc network (VANET) broadcasts a traffic-related message to nearby terminals within a time range of 100 to 300 $msec$ [2]. This message contains information regarding the vehicle's location, speed, heading, etc [1]. Considering the open nature of wireless vehicular communication, VANETs are susceptible to passive and active attacks such as interception, fabrication, and modification [3]. These attacks can be avoided by authenticating the received message to determine the legitimacy of the sender [4]. Generally, a VANET architecture involves a trusted authority (TA), roadside units (RSUs), and vehicles' wireless communication devices known as "onboard units" (OBUs) [3].

Conventional approaches of authentication in VANETs have traditionally been investigated using elliptic curve cryptosystem (ECC) [5], [6], bilinear pairing (BP) [7]–[9], and hashing operations. Many factors have to be considered when developing an authentication scheme, including latency, complexity, security, and privacy. The state-of-the-art for authentication is divided into three categories: public key infrastructure (PKI), identity (ID), and group signature (GS)-based schemes [3]. For PKI-based schemes, each transmission is accompanied by a digital certificate which certifies that the attached public key belongs to the sender [3]. However, attaching a certificate to each transmission creates a significant communication overhead. Furthermore, a large storage capacity is needed to store a large number of digital certificates (~ 43800 [1]). For ID-based schemes, messages are verified by the recipient using the sender's public key and signed by the terminal's private key [5]. Nevertheless, generating and verifying signatures based on public key cryptography is computationally expensive [10]. For GS-based schemes, group members sign messages anonymously, supporting identity anonymity [7]. However, existing GS-based schemes suffer from high computation and communication costs for generating and distributing group public and private keys. In addition, these keys must be updated periodically to provide forward and backward secrecy. To overcome these limitations, blockchain-based authentication has emerged in recent studies. In [11], Otoum et al. introduce a Federated Learning-based framework for authenticating transactions in a decentralized pattern. Lu et al. [12] employed the blockchain to develop a proof of revocation and issuance of certificates. Son et al. [13] proposed a consortium blockchain-based V2I handover authentication protocol. Nevertheless, developing a reliable group key distribution method remains challenging, particularly in high mobility and dense traffic environments.

The following is a summary of the paper's contributions.

- We propose a blockchain-based group key distribution method that enables the RSU as a group manager to distribute and update the group session key between group members with minimum communication and computation costs using a smart contract. Accordingly, we develop a lightweight GS-based message authentication process.
- The smart contract's functionality is evaluated by implementing its built-in functions and measuring its associated gas costs using Ethereum's main network ($MainNet$).
- Besides security analysis, the proposed scheme is extensively compared to conventional approaches to prove its superiority in reducing the computation and communication costs of verifying and transmitting messages.

The paper is organized as follows. Section II describes the proposed scheme. Sections III and IV evaluate the security and performance aspects. Finally, Section V concludes the paper.

II. The Proposed Scheme

This section goes into detail about the system and scheme modeling. For simplicity, Table I lists the scheme's notations.

This work was supported by the Egyptian Government and a grant funded by the Egyptian Ministry of Defence.

M. Shawky, A. Jabbar, M. A. Imran, Q. H. Abassi, S. Ansari, and A. Taha are with the Communication Sensing and Imaging group, James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK (m.shawky.1@research.gla.ac.uk, [Abdul.Jabbar, Muhammad.Imran, Qammer.Abbasi, Shuja.Ansari, Ahmad.Taha]@glasgow.ac.uk).

M. Usman is with the School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow, G4 0BA, UK (muhammad.usman@gcu.ac.uk)

TABLE I: Notations

Symbol	Definition
Sk_{TA}, Pk_{TA}	The system secret and private keys, respectively
Sk_{RSU_k}, Pk_{RSU_k}	RSU_k's private and public keys, respectively
Sk_{V_i}, Pk_{V_i}	V_i's private and public keys, respectively
K_{GS}, C_{GS}	The group session key and its encrypted parameter
SC_{RSU_k}	The smart contract deployed by RSU_k
$SCID_{RSU_k}$	The smart contract's address
T_i, T_r	σ_i's timestamp and receiving time, respectively
T_Δ	The freshness expiry period [00:00:59]
PID_{V_i}	The pseudo-identity of the vehicle V_i
$Cert$, T_R	The terminal's digital certificate and its expiry date

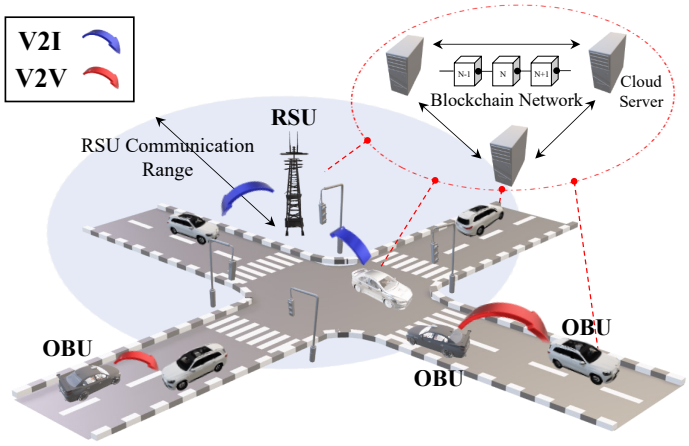


Fig. 1: System modeling.

Algorithm 1: Smart Contract for **GSKdistribution**

Given: function name, parameter settings

Require: Setting up functions

struct V2V {uint PID; uint CGS;
} //Defining the types of the input parameters
address RSU = $0xcbb21012b86b594223E43FB9c5017662$
$4F357463b$; //Defining the address of the RSU
mapping (uint → uint256) private PID2TX;//Defining a local function "PID2TX" that maps PID_{V_i} to $TxID$
function Deployer ( ) public {RSU = msg.sender;
} //Defining the SC's deployer as the RSU
modifier onlyowner {require (msg.sender == RSU);
_;}//Only the RSU can successfully run the **Deployer** function
V2V GSKdistribution1;
function IssueGSK (uint $\_PID$, uint $\_CGS$)
onlyowner public returns (uint, uint) {
GSKdistribution1.PID = $\_PID$;
GSKdistribution1.CGS = $\_CGS$;
return (GSKdistribution1.PID, GSKdistribution1.CGS)
} //Publishing a transaction Tx by the owner "RSU", which contains C_{GS} associated with PID_{V_i} and retrieving $TxID$
function UpdateGSK (uint PID, uint256 $TxID$)
onlyowner public {PID2TX [PID] = $TxID$;
} //The owner "RSU" maps PID_{V_i} to $TxID$.
function ViewGSK (uint PID) public view returns (uint256) {return PID2TX [PID];
} //This function can be invoked by any vehicle V_i using its corresponding PID_{V_i} to retrieve $TxID$

A. System modeling

The network comprises the following entities - see Fig. 1.

1) Trusted authority (TA): The TA is a trusted third party that initialises the system's parameters and registers network terminals. It is the only terminal that holds the link between the vehicle's real identity and its digital certificate. It also can expose the vehicle's real identity in case of misbehaving (i.e., constructing attacks or driving an unregistered vehicle).

2) Roadside units (RSUs): The RSU_k serves as a group manager responsible for the enrolment/revocation of vehicles getting in/out from its coverage area. In addition, it updates the group session key K_{GS} dynamically to ensure forward and backward secrecy. It is able to deploy and interact with its smart contract SC_{RSU_k} through transactions in the blockchain.

3) Vehicles' onboard units (OBUs): Each vehicle serves as a group member in a specific region and has a wireless communication device to communicate with surrounding vehicles. It is also capable of accessing the blockchain network and invoking the **ViewGSK** function using SC_{RSU_k}.

4) Smart contract-based blockchain: Smart contracts are code-based digital agreements whose terms and conditions are published on the decentralized blockchain network via transactions and written in the *Solidity* programming language. Algorithm (1) presents the smart contract of the K_{GS} distribution process with a command-by-command explanation. In the proposed smart contract, four functions are involved: **Deployer**, **IssueGSK**, **UpdateGSK**, and **ViewGSK**. The **Deployer**( ) function is used to define the owner of the smart contract SC. In our scenario, the RSU_k in each region acts as the owner and the deployer of the SC_{RSU_k}. The **IssueGSK**$(uint\_PID, uint\_CGS)$ function can only be invoked by the owner RSU_k to publish a transaction Tx contains the encrypted group key C_{GS} associated to the vehicle V_i's pseudo-identity PID_{V_i}, retrieving the published Tx's address $TxID$. Similarly, the **UpdateGSK**$(uint\_PID, uint\_TxID)$ function can only be invoked by the owner RSU_k and it is used to map the retrieved $TxID$ to PID_{V_i}. At last, the **ViewGSK**$(uint\_PID)$ function is invoked by V_i to retrieve $TxID$ related to PID_{V_i}. Using $TxID$, V_i obtains Tx's contents, C_{GS}, from the blockchain.

B. Scheme modeling

The proposed blockchain-based group signature scheme involves four phases, i.e., initialisation, registration, group session key generation, signature generation and verification.

1) Initialisation phase: TA performs the following steps to initialise the system's public and private parameters.

- TA chooses two prime numbers, p and q, with a length of 160 $bits$ used to initialise the elliptic curve $E : y^2 = x^3 + ax + b \ mod \ p$, where $(a, b) \in Z_q^*$ in a condition of $\Delta = 4a^3 + 27b^2 \neq 0$.
- TA chooses the generator g of length q and creates the cyclic additive group $\mathbb{G}$ that combines all points on E along with the infinity point $\mathcal{O}$.
- TA randomly chooses the system secret key $Sk_{TA} \in Z_q^*$, then computes its related public parameter $Pk_{TA} = Sk_{TA}.g$. In addition, TA chooses the SHA-256 hash function $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{N_1}$, where $N_1 = 256$ $bits$.
- Finally, the public parameters are $PPs = \langle a, b, p, q, g, Pk_{TA}, H_1 \rangle$.

2) Registration phase: TA performs the following steps to register all network terminals.

- For each RSU, TA publishes RSU_k's smart contract SC_{RSU_k} and retrieves its associated address

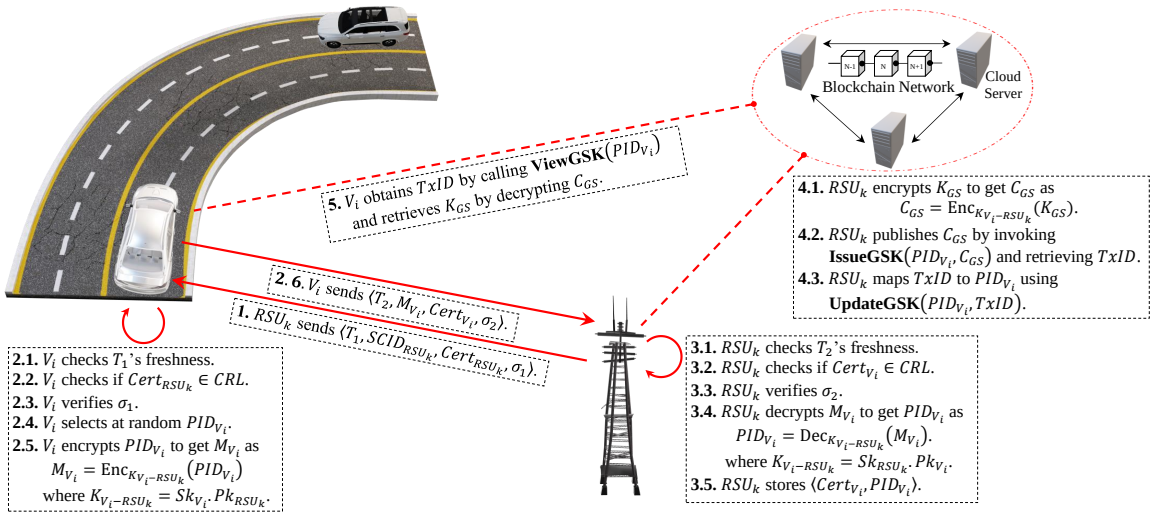


Fig. 2: Group session key distribution process.

$SCID_{RSU_k}$. After that, TA chooses the RSU_k's private key $Sk_{RSU_k} \in Z_q^*$ and computes its related public parameter $Pk_{RSU_k} = Sk_{RSU_k}.g$. Then, TA generates RSU_k's long term digital certificate $Cert_{RSU_k} = \langle Pk_{RSU_k}, T_R, \sigma_{TA} \rangle$, where T_R is the expiry date and $\sigma_{TA} = \text{Sign}_{Sk_{TA}}(Pk_{RSU_k} \| T_R)$. At last, TA stores $\langle PPs, Sk_{RSU_k}, Cert_{RSU_k}, SCID_{RSU_k} \rangle$ onto RSU_k.

- As for each vehicle V_i, TA checks the V_i's real identity RID_{V_i}, chooses the V_i's private key $Sk_{V_i} \in Z_q^*$ and computes its related public parameter $Pk_{V_i} = Sk_{V_i}.g$. Then, TA generates V_i's long term digital certificate $Cert_{V_i} = \langle Pk_{V_i}, T_R, \sigma_{TA} \rangle$, where $\sigma_{TA} = \text{Sign}_{Sk_{TA}}(Pk_{V_i} \| T_R)$. At last, TA stores $\langle PPs, Sk_{V_i}, Cert_{V_i} \rangle$ onto V_i.

3) Group session key generation phase: As shown in Fig. 2, this phase comprises the following steps:

- *Step 1*: In each region, there is a RSU_k that periodically broadcasts an enrollment message in the form of $\langle T_1, SCID_{RSU_k}, Cert_{RSU_k}, \sigma_1 \rangle$, where T_1 is the timestamp and the signature $\sigma_1 = \text{Sign}_{Sk_{RSU_k}}(T_1 \| SCID_{RSU_k} \| Cert_{RSU_k})$.
- *Step 2*: For each vehicle V_i in the communication range of the RSU_k, V_i checks T_1's freshness by finding out if $T_r - T_1 \leq T_\Delta$ holds or not to avoid replay attacks, verifies the signature σ_1 as $\text{Verf}_{Pk_{RSU_k}}(\sigma_1)$ to avoid impersonation attacks, and checks if $Cert_{RSU_k} \in CRL$. Then, V_i replies with a message in the form of $\langle T_2, M_{V_i}, Cert_{V_i}, \sigma_2 \rangle$, where $M_{V_i} = \text{Enc}_{K_{V_i-RSU_k}}(PID_{V_i})$, $K_{V_i-RSU_k} = Sk_{V_i}.Pk_{RSU_k}$, PID_{V_i} is a random number $\{0,1\}^{N_2}$ of length $N_2 = 256$ bits chosen by V_i, and $\sigma_2 = \text{Sign}_{Sk_{V_i}}(T_2 \| M_{V_i} \| Cert_{V_i})$.
- *Step 3*: The RSU_k in turn checks T_2's freshness, verifies the signature σ_2 as $\text{Verf}_{Pk_{V_i}}(\sigma_2)$, checks if $Cert_{V_i} \in CRL$, then decrypts M_{V_i} to get PID_{V_i} as $\text{Dec}_{K_{V_i-RSU_k}}(M_{V_i})$, where $K_{V_i-RSU_k} = Sk_{RSU_k}.Pk_{V_i}$ (using Diffie-Hellman key exchanging protocol). At last, RSU_k stores $Cert_{V_i}$ and its associated PID_{V_i}.
- *Step 4*: The RSU_k encrypts the group session key K_{GS} to get $C_{GS} = \text{Enc}_{K_{V_i-RSU_k}}(K_{GS})$ and uses the **IssueGSK**(PID_{V_i}, C_{GS}) function to publish C_{GS} related to PID_{V_i} through a transaction Tx. At last, RSU_k maps the transaction address $TxID$ to PID_{V_i} using the **UpdateGSK**$(PID_{V_i}, TxID)$ function.
- *Step 5*: Finally, V_i retrieves $TxID$ by calling the **ViewGSK**(PID_{V_i}) function using $SCID_{RSU_k}$. By using $TxID$, V_i can obtain the transaction Tx information, including C_{GS}. At last, V_i decrypts C_{GS} to get K_{GS} as $\text{Dec}_{K_{V_i-RSU_k}}(C_{GS})$.

Note that the SC's **IssueGSK** and **UpdateGSK** functions allow the RSU_k to dynamically update K_{GS} of group members without incurring an additional communication cost.

4) Signature generation and verification phase: In this phase, the signature is generated by V_i and verified by the group members V_j (i.e., surrounding vehicles) $\forall j \in [1, N-1]$, where N is the total number of vehicles in the communication range of RSU_k. This phase is presented in a two-step process.

- *Step 1*: V_i broadcasts a safety-related message m to surrounding vehicles in the form of $\langle m, T_3, PID_{V_i}, \sigma_3 \rangle$, where $\sigma_3 = \text{Enc}_{K_{GS}}(H_1(m \| T_3 \| PID_{V_i}))$.
- *Step 2*: $\forall j \in [1, N-1]$, V_j checks T_3's freshness and verifies σ_3 by testing if $H_1(m \| T_3 \| PID_{V_i}) \stackrel{?}{=} \text{Dec}_{K_{GS}}(\sigma_3)$ holds or not.

III. Security Analysis

In this section, we demonstrate that the proposed scheme complies with VANET security and privacy requirements.

A. Message authentication

The proposed scheme allows the group manager RSU_k to initially authenticate V_i using TA's signature $\sigma_{TA} \in Cert_{V_i}$, which proves V_i's ownership to Pk_{V_i}. Thus, it is hard to forge a valid signature signed by Sk_{V_i} under the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP). While V_j verifies V_i's signature for subsequent transmissions by checking whether $H_1(m \| T_i \| PID_{V_i}) \stackrel{?}{=} \text{Dec}_{K_{GS}}(\sigma_i)$ holds,

wherein σ_i's security level depends on the key length $|K_{GS}|$ used for generating σ_i using symmetric key cryptography.

B. Conditional privacy/Identity anonymity

Conditional privacy is maintained since only TA retains the link between $Cert_{V_i}$ and RID_{V_i}, preventing the identification of RID_{V_i} by any other terminals inside the network.

C. Unlinkability

The proposed SC_{RSU_k} supports unlinkability as no terminal can link between $Cert_{V_i}$ and the dynamically updated PID_{V_i} since PID_{V_i} is sent encrypted to RSU_k and published decrypted in the blockchain, making it infeasible to track V_i's transmitted messages from different sessions.

D. Resistance to active attacks

This scheme proves to be resistant to the following attacks.

1) Resistance to modification: To modify the message contents, an attacker needs to forge a valid signature which is infeasible without having the group session key K_{GS}. Therefore, the recipient can easily detect modification attacks by verifying the attached signature $H_1(m\|T_i\|PID_{V_i}) \stackrel{?}{=} \text{Dec}_{K_{GS}}(\sigma_i)$.

2) Resistance to impersonation: To impersonate V_i, an attacker needs to generate a valid signature using Sk_{V_i} at the first transmission slot. In other words, the attacker needs to deduce Sk_{V_i} from $Pk_{V_i} = Sk_{V_i}.g$ under the difficulty of solving the ECDLP.

3) Resistance to replaying: The attached timestamp T_i allows the recipient to verify the received messages' freshness in the same session by checking if $T_r - T_i \leq T_\Delta$ holds. While the dynamically updated K_{GS} resists replaying attacks from different sessions, supporting forward and backward secrecy.

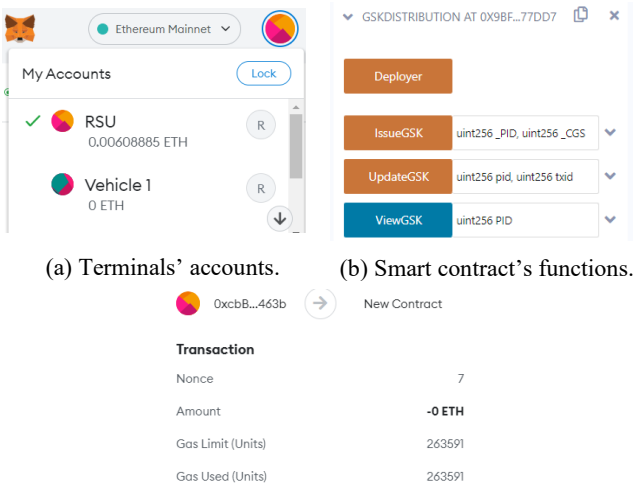
IV. Performance Analysis

This section evaluates the performance of the proposed scheme by implementing it on the Ethereum blockchain and measuring its computation and communication costs.

A. Implementation in the Ethereum blockchain

To discuss the feasibility of our scheme, we implement SC_{RSU_k} on the $Remix$ 0.25.4 compiler, an open-source smart contracts-based blockchain system. $Remix$-based smart contracts are written in $Solidity$, a javascript-like language. Using the $MeteMask$, a $chrome$ plug-in extension, we deploy and interact with SC_{RSU_k}'s functions in the $Ethereum\ MainNet$. Following are the details of the implementation.

1) In $MeteMask$, we set up two accounts with different addresses for the RSU_k and V_i, as shown in Fig. 3(a). Then, we charge the RSU_k's account and deploy the smart contract SC_{RSU_k} in the blockchain network, retrieving its associated address $SCID_{RSU_k} = 0xCF180843dA8E6fe5Ae3F7baE982e62640d430C$. Fig. 3(b) shows SC_{RSU_k}'s functions. More details about the deployment are given in Fig. 3(c), including the gas cost of deploying SC_{RSU_k}.
2) We simulate that RSU_k generates C_{GS} and publishes it using the **IssueGSK** function, invoking the published transaction's address $TxID$. At last, the RSU_k maps $TxID$ to PID_{V_i} using the **UpdateGSK** function.
3) Switching to V_i's account and using $SCID_{RSU_k}$, V_i obtains $TxID$ by calling the **ViewGSK** function. At last, V_i retrieves Tx contents, C_{GS}, from the blockchain using $TxID$.



(a) Terminals' accounts. (b) Smart contract's functions.

(c) The cost of deploying the smart contract.

Fig. 3: Terminals' addresses and SC_{RSU_k}'s functions.

In Table II, we show the gas costs per Wei for SC_{RSU_k}'s functions, where Wei is the smallest unit in ETH, 1 ETH $= 10^{18}$ Wei. It is noteworthy to mention that the **Deployer** function is the most expensive in terms of gas costs. Since this process is only performed once, it is relatively inexpensive. As for the actual costs of **IssueGSK**, **UpdateGSK**, and **ViewGSK** functions, these are 0.0024, 0.0004, and 0.0003 ETH, respectively, which is acceptable for group key distribution.

B. Computation and communication comparisons

This subsection shows a detailed analysis of computation and communication costs.

1) Computation comparison: We use the same estimates of time costs for different cryptographic operations in [13], see Table III. This evaluation is based on the MIRACL cryptographic library [14] using a quad-core i7 system with 16GB RAM. Based on that, we evaluate the time taken to verify a number of n received messages by the schemes presented in [5], [6], [9], and Ours, see Table IV. As can be seen, [5], [6], and [9] take $\approx (4.489n + 2.97)$, $(2.992n + 2.978)$, and $(7.62n + 5.042)$ $msec$, respectively to verify n messages. While the proposed scheme costs $\approx 0.004n$ $msec$. Fig. 4(a) shows the time taken to verify 1000 messages. In comparison with [5], [6], and [9], the proposed scheme is more computationally efficient since [5], [6], and [9] are public-key cryptography-based, whereas Ours is a symmetric key cryptography-based.

2) Communication comparison: For the evaluation of communication costs, we define different parameters' lengths. For ECC's parameters of curve type $y^2 = x^3 + ax + b\ mod\ p$, the

TABLE II: Gas costs associated with SC_{RSU_k}'s functions

Function	Gas used (Wei)	Actual cost (ETH)
Deployer	263591	0.002445
IssueGSK	69064	0.00049
UpdateGSK	46594	0.000308
ViewGSK	No fees	No fees

TABLE III: The average execution time of different cryptographic operations in $msec$ [13]

Operation	Definition	Time
T^{bp}	The bilinear pairing operation $e(.,.)$ in $\mathbb{G}_1$	13.44
T^{sm}_{bp}	The scalar multiplication operation in $\mathbb{G}_1$	2.521
T^{pa}_{bp}	The point addition operation in $\mathbb{G}_1$	0.018
T^{sm}_{ecc}	The scalar multiplication operation in $\mathbb{G}$	1.489
T^{pa}_{ecc}	The point addition operation in $\mathbb{G}$	0.008
T_h	The hashing operation (SHA-256)	0.003
T^{enc}_{AES}	Encryption operation using the AES algorithm	0.002
T^{dec}_{AES}	Decryption operation using the AES algorithm	0.001

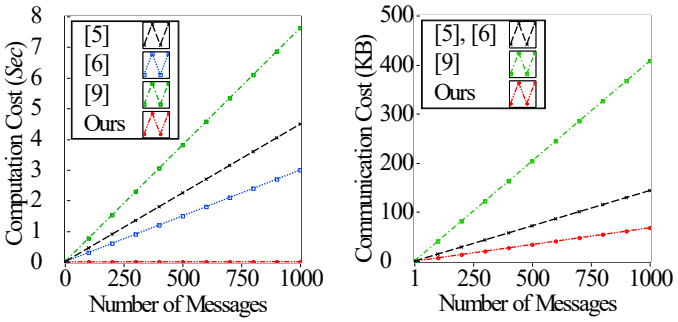
TABLE IV: Computation and communication comparisons

Scheme	Verification cost	Transmission cost
[5]	$(3n+2)T^{sm}_{ecc} + (2n-1)T^{pa}_{ecc} + (2n)T_h \approx 4.489n + 2.97\ msec$	$144n\ bytes$
[6]	$(2n+2)T^{sm}_{ecc} + (n)T^{pa}_{ecc} + (2n)T_h \approx 2.992n + 2.978\ msec$	$144n\ bytes$
[9]	$(3n+2)T^{sm}_{bp} + (3n)T^{pa}_{bp} + (n)T_h \approx 7.62n + 5.042\ msec$	$408n\ bytes$
Ours	$(n)T_h + (n)T^{dec}_{AES} \approx 0.004n\ msec$	$68n\ bytes$

length of an element in $\mathbb{G}$ and Z^*_q are 40 and 20 bytes, respectively. For BP's parameters of curve type $y^2 = x^3 + x\ mod\ p$, the length of an element in $\mathbb{G}_1$ and Z^*_q are 128 and 20 bytes, respectively. While the length of the hashed value and timestamp are 32 and 4 bytes, respectively. According to [5], $\langle PID^1_i, PID^2_i, R_i, T_i, \sigma_m \rangle$ represents the signature, where $\{PID^1_i, PID^2_i, R_i\} \in \mathbb{G}$, $\sigma_m \in Z^*_q$, and T_i is the timestamp. Thus, the total signature size is $(3 \times 40 + 4 + 20) = 144$ bytes. Similarly, we calculated the transmission costs of [6] and [9], as presented in Table IV. According to Ours, $\langle T_i, PID_{V_i}, \sigma_i \rangle$ represents the signature, where PID_{V_i} and σ_i have the same length, 32 bytes each, and T_i is the timestamp. Thus, the total signature size is $(2 \times 32 + 4) = 68$ bytes. Fig. 4(b) shows the communication cost of transmitting 1000 messages. From Fig. 4(b), we conclude that the proposed scheme saves high communication costs over those in [5], [6], and [9].

V. Conclusions

This paper proposes a blockchain-based group key distribution technique that exploits the immutability of blockchain technology to distribute group session keys among group members via a smart contract. The smart contract's functions enable the group manager to distribute and update the group key in a secure manner without violating VANET's security or privacy requirements. The scheme was tested for its resistance to active attacks. Additionally, the computation comparison demonstrated that the proposed scheme reduces the time



(a) Computation comparison. (b) Communication comparison.

Fig. 4: Computation and communication costs of verifying and transmitting a number of n messages.

needed to verify 1000 messages by 99% when compared to that of [5], [6], and [9]. While the transmission cost is reduced by 52.7% and 83.3% compared to that of [5], [6] and [9], respectively. Our future research will include examining how physical layer-based key extraction can be used to design a dynamic message authentication scheme for VANETs.

References

[1] M. A. Al-Shareeda, M. Anbar, and I. H. Hasbullah, "Survey of Authentication and Privacy Schemes in Vehicular Ad Hoc Networks", *IEEE Sensors Journal*, vol. 21, no. 2, Jan. 2021.

[2] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States", in Proceedings of the IEEE, vol. 99, no. 7, pp. 1162-1182, Jul. 2011.

[3] I. Ali, A. Hassan, and F. Li, "Authentication and Privacy Schemes for Vehicular Ad Hoc Networks (VANETs): A Survey", *Vehicular Communications*, vol. 16, pp. 45-61, Apr. 2019.

[4] D. D. N. Nguyen, K. Sood, M. R. Nosouhi, Y. Xiang, L. Gao, and L Chi, "RF Fingerprinting-Based IoT Node Authentication Using Mahalanobis Distance Correlation Theory", *IEEE Net. Let.*, vol. 4, no. 2, Jun. 2022.

[5] D. He, S. Zeadally, B. Xu, and X. Huang, "An Efficient Identity-based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks", *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2681–2691, Dec. 2015.

[6] J. Li, K. -K. R. Choo, W. Zhang, et al., "EPA-CPPA: An Efficient, Provably-Secure and Anonymous Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks", *Journal of Vehicular Communications*, vol. 13, Jul. 2018.

[7] C. Zhang, X. Xue, L. Feng, X. Zeng, and J. Ma, "Group-Signature and Group Session Key Combined Safety Message Authentication Protocol for VANETs", *IEEE Access*, vol. 7, pp. 178310-178320, Jan. 2019.

[8] H. Liu, H. Wang, and H. Gu, "HPBS: A Hybrid Proxy based Authentication Scheme in VANETs", *IEEE Access*, vol. 8, Sep. 2020.

[9] J. Li, Y. Ji, K. -K. R. Choo, and D. Hogrefe, "CL-CPPA: Certificate-Less Conditional Privacy-Preserving Authentication Protocol for the Internet of Vehicles", *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10332-10343, Dec. 2019.

[10] G. Bansal, and Biplab Sikdar, "Location Aware Clustering: Scalable Authentication Protocol for UAV Swarms", *IEEE Net. Let.*, vol. 3, no. 4, Dec. 2021.

[11] S. Otoum, I. A. Ridhawi, and H. Mouftah, "A Federated Learning and Blockchain-enabled Sustainable Energy-Trade at the Edge: A Framework for Industry 4.0", *IEEE Internet of Things Journal*, Jan. 2022.

[12] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A Privacy-Preserving Trust Model Based on Blockchain for VANETs", *IEEE Access*, vol. 6, pp. 45655-45664, Aug. 2018.

[13] S. Son, J. Lee, Y. Park, Y. Park, and A. K. Das, "Design of Blockchain-Based Lightweight V2I Handover Authentication Protocol for VANET", *IEEE Trans. on Net. Sci. and Eng.*, vol. 9, no. 3, Jun. 2022.

[14] MIRACL Crypto Library: Multiprecision Integer and Rational Arithmetic C/C++ Library. Available: https://github.com/miracl/MIRACL.