Hindawi
Security and Communication Networks
Volume 2021, Article ID 9942632, 17 pages
https://doi.org/10.1155/2021/9942632

Research Article

V-Lattice: A Lightweight Blockchain Architecture Based on DAG-Lattice Structure for Vehicular Ad Hoc Networks

Xiaodong Zhang,[1,2] **Ru Li**,[1,2] **Wenhan Hou**,[1,2] **and Hui Zhao**[1,2]

[1]*Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing, Hohhot 010021, China*
[2]*College of Computer Science, Inner Mongolia University, Hohhot 010021, China*

Correspondence should be addressed to Ru Li; csliru@imu.edu.cn

Received 28 March 2021; Revised 19 April 2021; Accepted 7 May 2021; Published 30 May 2021

Academic Editor: Yuanlong Cao

With the development of wireless communication technology and the automobile industry, the Vehicular Ad Hoc Networks bring many conveniences to humans in terms of safety and entertainment. In the process of communication between the nodes, security problems are the main concerns. Blockchain is a decentralized distributed technology used in nonsecure environments. Using blockchain technology in the VANETs can solve the security problems. However, the characteristics of highly dynamic and resource-constrained VANETs make the traditional chain blockchain system not suitable for actual VANETs scenarios. Therefore, this paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. In V-Lattice, each node (vehicle or roadside unit) has its own account chain. The transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. At the same time, in order to encourage more nodes to participate in the blockchain, a reputation-based incentive mechanism is introduced in V-Lattice. This paper uses Colored Petri Nets to verify the security of the architecture and verifies the feasibility of PoW anti-spam through experiment. The validation results show that the architecture proposed in this paper is security, and it is feasible to prevent nodes from generating malicious behaviors by using PoW anti-spam.

1. Introduction

With the development of wireless communication technology and the automobile industry, the Vehicular Ad Hoc Networks (VANETs) have developed significantly, which brings many conveniences to humans in terms of safety and entertainment. Through the collaboration of Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrians (V2P), and Vehicle-to-Cloud (V2C) communication, VANETs can enhance driving safety and traffic efficiency and realize Intelligent Traffic System (ITS) better. In ITS, a lot of sensor data need to be transferred between nodes. In particular, video data collected by cameras, such as traffic accident information and road condition information, need to be transmitted to other nodes through mobile video communication technology. However, in the process of communication between the nodes in the VANETs, security and privacy are the main concerns [1], including the security protection issues (such as integrity and correctness) of the VANETs data in transmission process, the security issues (such as non-tampering) of VANETs data stored in the data center, the access control issues (such as identity verification), and privacy protection issues.

The traditional approach to solve security problems is based on centralized approach which requires a trusted central entity. However, through a trusted central entity, there is a single point of failure problem. Moreover, the data stored in a trusted central do not have a technical method to ensure its security (nontampering, traceability), and they are only guaranteed at the legal level. There are also some approaches based on the distributed approach, but the lack of trust between distributed entities makes it difficult to implement in actual environments.

Blockchain is a distributed technology. It uses cryptography and hash functions to store data in a chain to ensure that data are tamper-resistant and traceable. And the technology uses a consensus protocol to ensure data consistency. At present, the use of blockchain is very extensive, including supply chain management, agriculture, Internet of Things (IoT), artificial intelligence (AI), and autonomous vehicles. It can be seen that using blockchain technology in the VANETs environment can solve the security problems and the dependence on trusted central entities.

At present, there have been many studies on using blockchain in the VANETs environment, mainly including the architecture [1–4], authentication mechanism [5–7], privacy protection [8–10], trust management [11–13], certificate management [14, 15], and data sharing [16–18]. These studies mainly focus on the architecture of the combination of blockchain and the VANETs and how to use the blockchain to solve the security issues in the VANETs environment. Moreover, they are all based on the traditional chain blockchain structure. However, the VANETs is self-organizing and highly dynamic. Moving vehicles will gather at intersections to form a multidomain network. The data sharing between V2V and V2I is mostly temporary and dynamic, which make it difficult to synchronize data between nodes and resulting in more forks and low consensus efficiency. At the same time, vehicles in the VANETs usually have limited resources. The compute and storage capabilities are usually not particularly strong. So, a complete blockchain cannot be run and stored on the vehicle. It can be seen that the characteristics of highly dynamic and resource-constrained of the VANETs make the traditional chain blockchain system not suitable for actual VANETs scenarios.

With the development of blockchain technology, the structure of the blockchain has also evolved from a traditional chain structure to a directed acyclic graph (DAG) structure. In the DAG structure, the constituent unit is a transaction which does not need to be packaged into blocks. Moreover, in the DAG structure, each transaction can point to multiple previous transactions, allowing the blockchain to generate forks. Therefore, the DAG structure can make transactions reach consensus parallelly, which greatly improves the speed of processing transactions, that is, TPS. It can be seen that through the DAG structure, the consensus efficiency and transaction throughput of the system can be improved, meanwhile the problem of low scalability can be solved. Currently, the researches using DAG structure include IOTA [19], Byteball [20], InterValue [21], Nano [22], and JURA [23]. These researches use DAG structure to build a blockchain that can run stably for a long time. It shows that the DAG structure can replace the traditional chain structure to show better performance. In particular, the DAG-lattice structure is used in both the Nano and JURA projects. In the DAG-lattice structure, each node account has its own account chain, and only the node itself can add transactions to its own account chain. So, the transactions between accounts can be added to the blockchain asynchronously and parallelly. Moreover, in the lattice DAG structure, the node can prune the transactions from the account chain of the node out of communication and only keep their latest transactions without affecting the system consensus. It can be seen that using the DAG-lattice structure in the VANETs environment can solve the problems caused by high dynamics and resource constraints.

Therefore, this paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. This architecture is general and suitable for highway and urban road. In this architecture, each node has its own account chain. The transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. At the same time, in order to incentivize vehicle and roadside unit (RSU) to participate in the blockchain network actively and without malicious intent, this paper uses a reputation-based incentive mechanism; that is, each node participating in the blockchain has a reputation score, and the reputation score of nodes can be updated dynamically based on the node's behavior. The main contributions to this paper are as follows.

(1) Based on the DAG-lattice structure, this paper proposes a lightweight blockchain architecture for VANETs. The lightweight nature is reflected in two aspects. One is the small amount of calculation. In this paper, the PBFT consensus algorithm is used, which is not discriminatory in computing power. The other is the small storage capacity. Vehicles with limited storage capacity store pruned blockchain instead of full blockchain.

(2) This paper introduces a reputation-based incentive mechanism to encourage more nodes to participate in the blockchain, and through updating the node's reputation score dynamically, it motivates normal nodes and punishes malicious nodes.

(3) This paper proposes a consensus method that can conduct asynchronous consensus on transactions generated by nodes.

(4) Aiming at the lightweight blockchain architecture for VANETs proposed, this paper describes how common applications in the VANETs work under this architecture.

The rest of the paper is organized as follows. Section 2 describes research on the architecture introducing blockchain technology in VANETs and the current using of the DAG-lattice structure; Section 3 introduces the relevant components of the architecture proposed in this paper; Section 4 introduces the proposed lightweight blockchain architecture in detail; Section 5 describes the malicious attack scenarios that may occur in this architecture and analyzes how this paper deals with these malicious attacks; Section 6 verifies the proposed architecture; finally, the full paper is summarized in Section 7.

2. Related Work

2.1. Research on the Architecture of Using Blockchain in VANETs. At present, there have been many researches on

proposing the security architecture for VANETs using blockchain. These studies mainly include two categories. One is that the nodes (vehicle and RSU) in VANETs do not participate in the blockchain, and the other is that the nodes in VANETs participate in the blockchain.

In the research case where the nodes in VANETs do not participate in the blockchain, the blockchain is deployed as a storage service in nodes or platforms which maintain the operation of the blockchain outside the VANETs. Rahman et al. [4] proposed a secure Internet of Vehicles (IoV) framework that allowed vehicle data to be stored in blockchain and off-chain repositories for secure sharing with one's community of interest. In this framework, the blockchain provides storage as a service. Xu et al. [24] introduced a new framework for secured intelligent vehicle data sharing, namely, biometric blockchain (BBC). In the BBC, each vehicle can be connected to the BBC-based platform and store their IV-BC through the BBC-based cloud service. Kulathunge and Dayarathna [25] proposed a VANET communication framework based on blockchain functions. In this architecture, data are stored in the Hyperledger Fabric blockchain, and the VANETs nodes communicate with the blockchain through the REST protocol.

However, the blockchain runs on nodes outside the VANETs as a distributed storage environment, which can only guarantee the security of the data in the storage process and cannot improve the security of the data in the communication process through blockchain technology. Therefore, researchers have proposed some architectures in which the nodes in the VANETs participate in the blockchain. In these architecture studies, the blockchain runs in vehicles or RSUs in the VANETs, and the operation of the blockchain is maintained by the vehicles or RSUs. Yuan and Wang designed a seven-layer conceptual model for ITS in [26] and proposed a B2ITS framework. Dorri et al. [27] proposed a blockchain-based distributed privacy protection and security architecture for smart vehicles. The nodes in the architecture are clustered, and only the cluster heads (CHs) are responsible for managing the blockchain and executing its core functions. Singh and Kim [28] proposed an Intelligent Vehicle-Trust Point (IV-TP) mechanism for IV communication among IVs using blockchain technology. In this mechanism, the IV-TP data are managed through the blockchain. Leiding et al. [29] proposed a transparent, self-managed, and decentralized system which combined Ethereum and VANETs. In this system, each entity has an Ethereum address, and RSU provides Ethereum-based applications which have been deployed to the Ethereum blockchain. Sharma et al. [2] proposed a vehicle network architecture based on blockchain in the smart city (Block-VN). Block-VN includes controller nodes, miner nodes, and ordinary nodes. The ordinary nodes can send service request messages to miner nodes (vehicles) or controller nodes. Jiang et al. proposed a blockchain-based distributed VANETs architecture in [30]. According to different application purposes, the blockchain is divided into 5 types. The various blockchains do not communicate with each other, and they also have 5 different types of blockchain nodes.

The current architecture researches are considered from the perspective of application. The use of blockchain can solve some security issues, privacy, and so on. Moreover, the proposed architectures mainly study that the blockchain runs on which entities, what information is stored on the entities, and how the entities interact to ensure security in the VANETs. However, they do not consider the high dynamics of the VANETs and the problems caused by the use of blockchain under the condition of limited node resources.

2.2. Usage of DAG-Lattice Structure. The DAG-lattice structure is a kind of DAG structure. The first to use this structure was the Nano project [22]. In this project, each account has its own blockchain. The sending transaction and receiving transaction are separated, which can provide almost instantaneous transaction speed and unlimited scalability. Moreover, the transaction tracks the account balance, so that blockchain can be pruned without affecting the performance and security. In view of the limitations of current blockchain technology and the scalability requirement of having millions of TPS in the future, the JURA team proposed the decentralized JURA [23], in which novel data structure Fusus, PoU consensus mechanism, verifiable random function technology, dynamic monitored and distributed sharding, and artificial intelligence is used. The essence of Fusus is the DAG-lattice structure. The transaction records of accounts form the lattice, and the transaction records of each account are organized by DAG structure.

Both Nano and JURA are cryptocurrencies based on blockchain. However, some researchers have used the DAG-lattice structure in a noncryptocurrency environment. Zhou et al. [31] used the DAG-lattice structure in data tokenization and proposed the lattice blockchain model, namely, DLattice. This model has a double-directed acyclic graph (Double-DAG) structure, and each account is composed of a Token-Chain and a Data-Tree.

At present, the DAG-lattice structure has not been used in the VANETs environment.

3. The Components of V-Lattice

This section defines the basic components of V-Lattice from four aspects: node, account, transaction, and ledger.

3.1. Node. The node is software that runs on entities in the blockchain network. It follows the relevant protocols of the blockchain and can run all or part of the blockchain-related operations, including generating transaction, verifying transaction, transaction consensus, and storing transaction. In the VANETs environment, the entities are mainly mobile vehicles and RSUs fixed on both sides of the road. When a blockchain node is running on a moving vehicle, the mobility of the vehicle causes the vehicle to frequently join or leave the blockchain network. Moreover, the moving vehicle often forms small clusters at intersections, which makes it difficult to maintain a unified ledger among vehicles; when the blockchain node is running in fixed RSUs, the RSUs are

fixed and can maintain a stable network. However, because the blockchain ledger is stored on the RSUs, it cannot be guaranteed to keep communicating with the RSUs during the movement of vehicles. When the vehicles cannot communicate with the RSUs through a single hop or multiple hops, it is impossible to obtain data from blockchain or synchronize the transactions to the RSUs in real time.

Therefore, both vehicles and RSUs participate in the blockchain network as blockchain node and jointly maintain the blockchain in V-Lattice. The node running on the vehicle is called vehicle node, denoted as $\mathrm{VNode}_i \in \{\mathrm{VNode}_1, \mathrm{VNode}_2, \ldots, \mathrm{VNode}_N\}$, where N is the number of vehicles. The node running on RSU is called RSU node, denoted as $\mathrm{RNode}_i \in \{\mathrm{RNode}_1, \mathrm{RNode}_2, \ldots, \mathrm{RNode}_M\}$, where M is the number of RSUs.

3.2. Account. The account is the identity of a blockchain node participating in the system. For ease of management, each blockchain node has only one account. Each account consists of a public key and private key pair $\{P_r, S_r\}$, where the public key P_r is publicly available on the entire network to identify each account, and the private key S_r is kept by the account itself and is not publicly disclosed.

The public and private key pair of the account is generated by the Trust Center (TC). The trusted center is an absolutely trusted infrastructure that will not be maliciously attacked and used for generating public and private key pair, authenticating identity, and issuing certificates. Before joining the blockchain network, vehicle nodes and RSU nodes firstly need to generate public and private key pair through the TC. The public and private key pair of the vehicle account is generated by the fixed attributes of the vehicle (vehicle license plate, vehicle type, vehicle manufacturer, vehicle owner information, etc.). The public and private key pair of the RSU account is generated by the fixed attribute (number, etc.) of the RSU.

The V-Lattice contains vehicle accounts and RSU accounts, which represent vehicle nodes and RSU nodes, respectively. The vehicle account is denoted as $\mathrm{VAC}_i \in \{\mathrm{VAC}_1, \mathrm{VAC}_2, \ldots, \mathrm{VAC}_N\}$, where N is the number of vehicles. RSU account is denoted as $\mathrm{RAC}_i \in \{\mathrm{RAC}_1, \mathrm{RAC}_2, \ldots, \mathrm{RAC}_M\}$, where M is the number of RSUs.

3.3. Transaction. The transaction is a series of operations that cause state change in the blockchain. In a traditional blockchain, multiple transactions need to be packaged into a block firstly. It takes a certain amount of time to pack a block, and the packed block is generally relatively large, which is not conducive to transmission in a highly dynamic and bandwidth limited VANETs environment. Therefore, a transaction is regarded as a block in V-Lattice. Transaction and block can be used interchangeably.

In the traditional blockchain, a chain is maintained between all accounts. One operation is a transaction, and a transaction may involve two accounts. However, for a blockchain with a lattice structure, each account maintains its own account chain. In order to reduce the coupling and enable transactions to operate asynchronously between account chains, a transaction under the traditional blockchain is separated into a sending transaction and a receiving transaction in V-Lattice, which is stored in sending and receiving account chains, respectively.

In the VANETs environment, the main behaviors of nodes include that node creates account, node broadcasts messages, node obtains messages from other nodes, and RSU updates node's reputation score. Therefore, the transaction types involved in V-Lattice include creating account transaction ($T_{\mathrm{acount\_create}}$), sending message transaction ($T_{\mathrm{message\_send}}$), sending reputation score transaction ($T_{\mathrm{reputation\_send}}$), and receiving transaction (T_{receive}).$T_{\mathrm{acount\_create}}$ is used to create the node's account; $T_{\mathrm{message\_send}}$ and T_{receive} appear in pairs to complete a message transmission together; $T_{\mathrm{reputation\_send}}$ and T_{receive} appear in pairs to complete a reputation score updating operation together. In particular, $T_{\mathrm{acount\_create}}$ only involves one account, so there is no need for T_{receive}.

3.4. Ledger. The ledger is the data maintained by all accounts, and each account has an account chain. For each account, the transaction content on the account chain includes message-related transactions and reputation-related transactions. In order to improve transaction processing speed, facilitate transaction query, and process transactions asynchronously, the V-Lattice divides the account chain into Message Chain (MC) and Reputation Chain (RC). The ledger is stored by the structure of a directed acyclic graph (DAG). The DAG-lattice structure of the ledger is shown in Figure 1.

The genesis block is generated by the system during initialization, and it is the parent block of all creating account transaction. $T_{\mathrm{acount\_create}}$ is the genesis transaction of the account chain. The MC and RC are, respectively, linked to $T_{\mathrm{acount\_create}}$. $T_{\mathrm{message\_send}}$ and T_{receive} exist in pairs, and $T_{\mathrm{message\_send}}$ must be quoted by the corresponding T_{receive}. Similarly, $T_{\mathrm{reputation\_send}}$ and T_{receive} exist in pairs, and $T_{\mathrm{reputation\_send}}$ must be quoted by the corresponding T_{receive}. The change of node's reputation score is caused by the behavior of the node. In this architecture, the behavior of the node refers to whether the node sends the correct message. In other words, the behavior of the node can be reflected by $T_{\mathrm{message\_send}}$. Therefore, $T_{\mathrm{reputation\_send}}$ must quote the corresponding $T_{\mathrm{message\_send}}$ that resulted in the reputation score changing.

Through the DAG-lattice structure shown in Figure 1, the transactions can be performed asynchronously between accounts. At the same time, the transactions on MC and RC can also be performed asynchronously within the same account.

4. Proposed V-Lattice

This section introduces the proposed V-Lattice in detail.

4.1. The Overview of V-Lattice. In V-Lattice, vehicles, and RSUs together form a blockchain network. When the

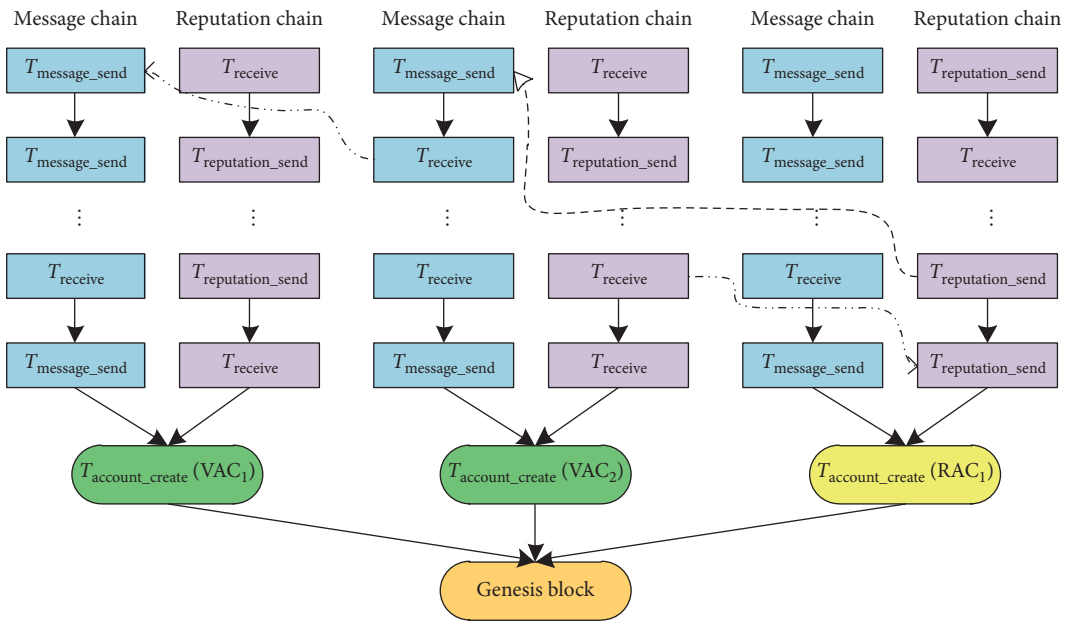

Figure 1: The DAG-lattice structure of ledger.

vehicles and the RSUs join the blockchain network, firstly they need to obtain the public and private key pair through the trusted center and then create account according to public key. The communication method between vehicles or between vehicles and RUSs is wireless communication, such as DSRC and 5G. The communication method between RUSs is wire communication. They communicate through the Gossip protocol to ensure the consistency of the blockchain. Figure 2 shows the network structure diagram.

The entities of this architecture mainly include Trust Center, vehicle, and RSU. The overall workflow chart between entities is shown in Figure 3.

Trusted Center, as an absolutely trusted entity, is used to generate public and private key pairs for vehicles and RSUs. Moreover, it is also used for assisting RSUs to verify the account information created by blockchain nodes.

Vehicle, as a lightweight node, has limited storage capacity and stores the pureed blockchain. In order to verify transactions and participate in consensus, the pureed blockchain stored in the vehicle should at least contain the creating account transaction and the latest message-related transactions and reputation-related transactions of the vehicle itself and other surrounding vehicles. However, as the vehicle is moving, the surrounding vehicles constantly change, which cause the vehicles to frequently update the locally pureed blockchain. Therefore, in order to prevent this problem and to improve the overall security of the blockchain, vehicles are encouraged to store more transactions.

RSU, as a full node, has the characteristics of large storage capacity and strong computing power. It can perform all blockchain functions and store all transaction records. The update of the reputation score requires a large amount of calculations, and the transaction information of other vehicles is required, so it is completed by the RSU. For the message sent by the node, the RSU needs to judge whether the message is true or false (the method of judging whether the message sent by the node is true or false is mainly realized through trust management in the VANETs. This part is not the focus of this paper and will not be discussed in detail). If it is found that a fake message is sent by a node, its reputation is reduced; otherwise, its reputation is increased. For nodes with high reputation score, there are more opportunities to generate blocks and get rewards. Meanwhile, the messages generated by them are processed firstly. For nodes with low reputation score, the messages will not be accepted by other nodes and they cannot get priority services. This can help encourage the vehicle to stay normal.

In V-Lattice, the operations performed by the vehicle include generating transactions, verifying transactions, forwarding transactions, and participating in consensus. The operations performed by RSU include generating transactions, verifying transactions, forwarding transactions, participating in consensus, and updating node's reputation score.

4.2. Transaction Structure

4.2.1. Creating Account Transaction. To create an account, the blockchain node needs to send a creating account transaction as shown in Table 1. The type field identifies the type of transaction, and its value is account_create. The account field indicates the address of the account to be created. The difficulty field is the difficulty of Proof of Work (PoW), and the nonce is a random number that meets the difficulty of PoW. The difficulty and nonce will be described in detail in Section 4.5. The timestamp field indicates the

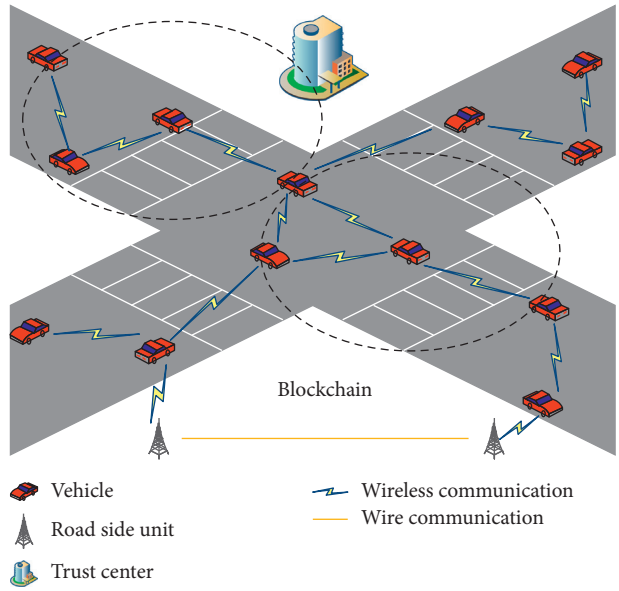

Figure 2: The network structure diagram.

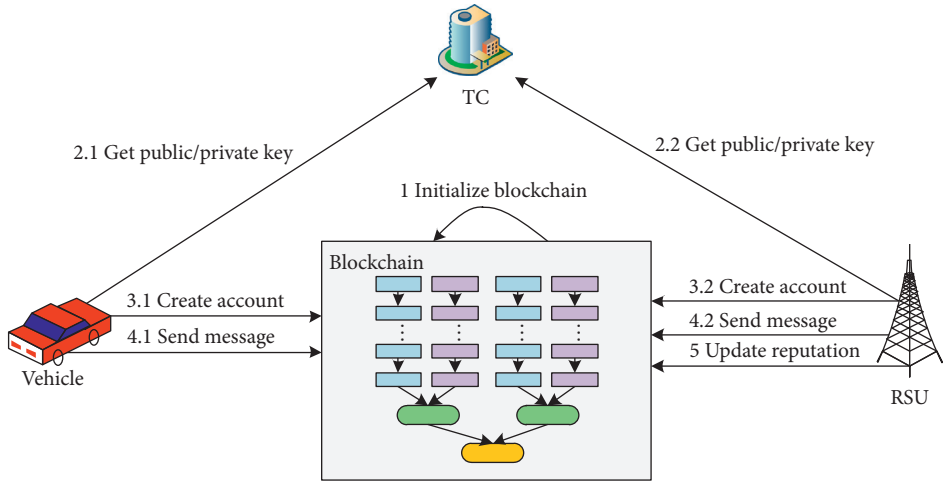

Figure 3: Overall workflow chart.

time when the transaction is created. The signature field is the signature of the transaction by the node creating transaction. Through signature, on the one hand, the node cannot deny its signature. On the other hand, it can also determine whether the transaction content has been tampered by malicious nodes, thereby ensuring the integrity of the information transmission process. The signature is generated by the transaction producer using his own private key S_r to encrypt the hash value of the transaction content (except the signature field), that is, signature = $E_{S_r}\{H_1(\text{type, account, nonce, difficulty, timestamp})\}$, where H_1 is the hash algorithm.

4.2.2. Sending Message Transaction. When the node sends message content, it firstly needs to send a sending message transaction as shown in Table 2. The type field identifies the type of transaction, and the value is message_send. The previous field identifies the hash value of the previous block. The source field is the address of the account that sent the transaction, and the destination field is the address of the account that receive the transaction. The msg field is used to indicate the message sent by node, including desc field and

Table 1: The structure of creating account transaction.

$T_{\text{acount\_create}}$ {
type: account_create,
account: f6b4c0685f. . .acc5f7918c,
nonce: 154883521,
difficulty: 0000000111. . .1111111111,
timestamp: 1606128999,
signature: af0c8bb4d5. . .b3786dd6ac
}

Table 2: The structure of sending message transaction.

$T_{\text{message\_send}}$ {
type: message _send,
previous: f69bb1178e. . .e5fdc9c7d1,
source: f6b4c0685f. . .acc5f7918c,
destination: 4d9c22f58f. . .7936b855da,
nonce:4565123156,
difficulty: 0000000111. . .1111111111,
timestamp: 1606133657,
msg: {desc, content {plaintext_cont, ciphertext_cont}},
signature: c8dbb6c772. . .6c46aabbf2
}

Table 3: The structure of sending reputation score transaction.

$T_{\text{reputation\_send}}$ {
type: reputation _send,
previous: 2174a3b1cd. . .5a57e16be9,
source: f6b4c0685f. . .acc5f7918c,
destination: 4d9c22f58f. . .7936b855da,
associate: d24c9bfe35. . .fcefebeb64,
nonce: 423426631123,
difficulty: 0000000111. . .1111111111,
timestamp: 1606133830,
reputation: {desc, score},
signature: 3af4cdace5. . .b326438866,
$\text{timestamp}_{\text{dest}}$: 1606133840,
$\text{signature}_{\text{dest}}$: b9bec112d8. . .63e65bd294
}

content field. The desc field is the description information of content, and content field refers to the specific message content to be sent which includes public content plaintext_cont field and encrypted content ciphertext_cont field. The desc field and plaintext_cont field are displayed in plaintext in the transaction, which can be directly read by other nodes, whereas the ciphertext_cont field is displayed in ciphertext in the transaction. Only after the sender authorizes it, it can be decrypted and read through the decryption key. The signature field is generated by the transaction producer using his own private key S_r to encrypt the hash value of the transaction content (except for the signature field), namely, signature = $E_{S_r}\{H_1$ (type, previous, source, destination, nonce, difficulty, timestamp, msg)}, where H_1 is the hash algorithm.

4.2.3. Sending Reputation Score Transaction. When RSU updates the reputation score of nodes, it needs to send a sending reputation score transaction as shown in Table 3 to the nodes. The type field identifies the type of transaction, and its value is reputation_send. The destination field indicates the account address of the node (vehicle or RSU) whose reputation score needs to be updated. The associate field is the hash value of the corresponding transaction that causes the node's reputation score to change. It is used to associate the change of node's reputation score with the node's behavior. The reputation field records the reputation score of the node, where desc field describes the update method of the node's reputation score, and the score field records the updated reputation score of the node. The $\text{timestamp}_{\text{dest}}$ field is the timestamp when the node whose reputation score is updated signs the transaction. The signature field and $\text{signature}_{\text{dest}}$ field, respectively, represent the signature of the transaction producer (i.e., the RSU updating the node's reputation score) and the node whose reputation score is updated. As the reputation score update is completed by the RSU, and the reputation score needs to be recorded on the Reputation Chain of the node whose reputation is updated, so the RSU and the node whose reputation is updated need to sign the transaction jointly. Through the joint signature method, the updated node can not only track the change of reputation score but also can prevent the updated node from modifying the reputation score, and at the same time can prevent the RSU from being attacked and maliciously updating the node's reputation score. The signature field is generated by the transaction producer using his own private key to encrypt the hash value of the transaction content (except for the signature field and $\text{signature}_{\text{dest}}$ field), that is, signature = $E_{S_r}\{H_1$ (type, previous, source, destination, associate, nonce, difficulty, timestamp, reputation)}, where S_r is the private key of the transaction producer and H_1 is the hash algorithm. The $\text{signature}_{\text{dest}}$ field means that the node whose reputation score is updated uses its own private key to encrypt the hash value of the transaction content (except for the signature field and $\text{signature}_{\text{dest}}$ field), that is, $\text{signature}_{\text{dest}} = E_{D^{S_r}}\{H_1$ (type, previous, source, destination, associate, nonce, difficulty, timestamp, reputation, $\text{timestamp}_{\text{dest}}$)}, where D^{S_r} is the private key of the node whose reputation is updated, and H_1 is hash algorithm.

4.2.4. Receiving Transaction. When node receives a sending message transaction or a sending reputation score transaction, it needs to send a receiving transaction as shown in Table 4. The type field is used to identify the type of the receiving transaction. If a receiving transaction is initiated for a sending message transaction, the type field is message_receive. And if a receiving transaction is initiated for a sending reputation score transaction, the type field is reputation_receive. The destination field indicates the address of the account that receives the transaction, which is consistent with the content of the source field. The associate field points to the hash value of the corresponding sending transaction and is used to associate the receiving transaction with the sending transaction. The signature field is generated by the transaction producer using his own private key to encrypt the hash value of the transaction content (except the signature field), that is, signature = $E_{S_r}\{H_1$ (type, previous, source, destination, associate, nonce, difficulty, timestamp)}, where S_r is the private key of the transaction producer, and H_1 is the hash algorithm. Only after the receiving transaction and the corresponding sending transaction are both agreed and added to the blockchain can an operation be considered complete.

4.3. Transaction Verification. After blockchain node receives the transaction T sent by the node K, ($K \in \text{VNode}_i \| \text{RNode}_i$),

Table 4: The structure of receiving transaction.

$T_{\text{receive}}\{$
type: reputation _receive/message _receive,
previous: 34676fa04c. . .97c2d3e2a2,
source: f6b4c0685f. . .acc5f7918c,
destination: f6b4c0685f. . .acc5f7918c,
associate: 1a6a65ede0. . .62d93672a1,
nonce: 551354523,
difficulty: 0000000111. . .1111111111,
timestamp: 1606133858,
signature: abf7727736. . .9854bae5d9
}

it needs to verify the transaction T. The verification content includes (1) whether the transaction is received for the first time; (2) the nonce value; (3) the timestamp; (4) the signature of the transaction; (5) the associated transaction. The specific verification procedures of transaction T are shown in Figure 4.

4.3.1. Whether the Transaction Is Received for the First Time. Upon receiving a transaction T, first it is necessary to verify whether the transaction is received for the first time. The verification method is to check the recent historical transaction information under the account that sent the transaction T to determine whether there is a copied transaction. If there is a copied transaction, that is, the transaction has been received before, it may be a copied transaction generated by a malicious node, and the transaction needs to be discarded.

4.3.2. The Nonce Value. The correctness of the nonce value can determine whether the blockchain node K has completed proof of work. Only when the node has completed the proof of work can it send transactions to the blockchain network. So, the verification of the correctness of the nonce value is very important. The method of verification is to combine the type field, the difficulty field, the timestamp field, and the nonce field of the transaction to calculate the hash value and determine whether the $\text{Hash}(\text{type}, \text{difficulty}, \text{timestamp}, \text{nonce})$ meets the PoW difficulty. If the nonce value meets the value of the difficulty field in the transaction, it means that the node has completed the corresponding proof of work; otherwise, the transaction is discarded.

4.3.3. The Timestamp. The purpose of verifying the timestamp is to determine whether the node K sends the transaction immediately after it generates the transaction and to prevent malicious nodes from generating precomputed POW attacks (detailed in Section 5.3). The method to verify the correctness of the timestamp is to verify whether the time internal Δt between the timestamp of the transaction timestamp or $\text{timestamp}_{\text{dest}}$ and the time of receipt of the transaction Time satisfies $\Delta t < \text{Threshold}_T$, where $\Delta t = \text{Time} - \text{timestamp} \| \text{timestamp}_{\text{dest}}$. If Δt exceeds the threshold Threshold_T, it means that the node does not send the transaction immediately and the verification failed. It should be noted that, for the sending reputation score transaction, it contains the timestamp and $\text{timestamp}_{\text{dest}}$. If the transaction is issued by the RSUs which update the reputation score, the timestamp will be verified. If the transaction is issued by the node whose reputation score is updated, then verify the $\text{timestamp}_{\text{dest}}$.

Under normal circumstances, the determinants of the time interval Δt include the PoW time t_T^{pow} of the transaction, the queue time t_T^{que} of the transaction in the node, and the transmission time t_T^{trans} from the sending node to the destination node of the transaction. Therefore, the threshold Threshold_T is expressed as follows:

$$\text{Threshold}_T = t_T^{\text{pow}} + t_T^{\text{que}} + t_T^{\text{trans}} + \eta_T. \tag{1}$$

t_T^{pow} is the time required for the node to complete PoW for the transaction. This time usually varies and is specifically related to the node's reputation score and computing power. However, the computing power between nodes is usually similar, so the main influencing factor of this time is the reputation score of the node.

t_T^{que} is the queuing time of transactions in the node, which is determined by the frequency of the sending packets and the frequency of generating transactions. If there is no malicious attack (e.g., Denial of Service), the frequency of transactions generated by the node is generally not too high.

t_T^{trans} is the transmission time of the transaction from the sending node to the receiving node, and this time is related to the network topology.

η_T is the error time. Since there may be error in the calculation of the times t_T^{pow}, t_T^{que} and t_T^{trans}, they need to be corrected by η_T. The value of η_T needs to be determined according to the real network environment and historical experience. If η_T is too small, it will cause normal transactions to fail to be verified. On the contrary, if the value of η_T is too large, transactions generated by malicious nodes may be mistaken for normal transactions.

4.3.4. The Signature of the Transaction. By verifying the signature of the transaction, it can be determined that the transaction is indeed generated by the producer of the transaction rather than forged by other malicious nodes. The method to verify the signature is firstly to calculate the hash value of the transaction (except for the signature field or $\text{signature}_{\text{dest}}$ field) $\text{hash}_1 = H_1(\text{type}, \text{account}, \text{nonce}, \text{difficulty}, \text{timestamp})$ (here we take the creating account transaction as an example), where H_1 is the hash algorithm, which is the same hashing algorithm as when signing. Then, it needs to use the public key P_r of the transaction producer to decrypt the signature and obtain the decrypted hash value $\text{hash}_2 = D_{P_r}\{\text{signature}\}$. Finally, it needs to judge whether hash_1 and hash_2 are the same. If $\text{hash}_1 = \text{hash}_2$, the signature

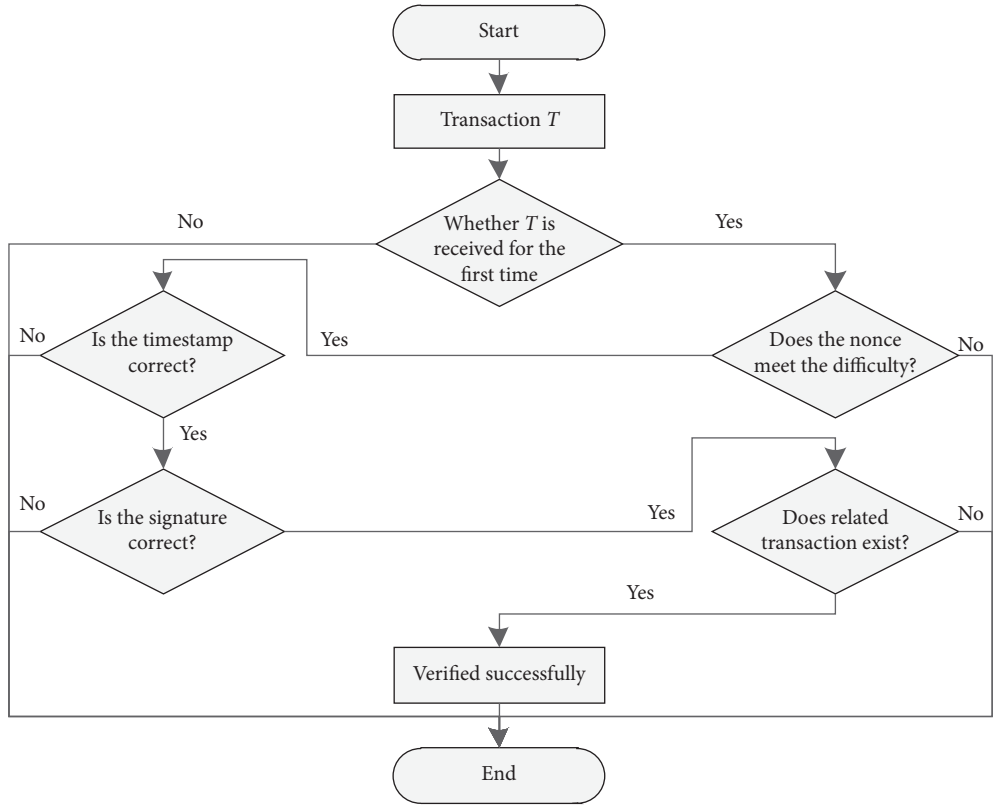

Figure 4: Transaction verification flow chart.

verification is passed; otherwise, the transaction is discarded. The specific verification process is as shown in Figure 5.

It is particularly important to note that, for the sending reputation score transaction, if the transaction is generated by the RSU that updates the node's reputation score, it only needs to verify the signature. If the transaction is sent by the node whose reputation score is updated, the signature and the $\text{signature}_{\text{dest}}$ need to be verified at the same time.

4.3.5. The Associated Transaction. For the creating account transaction and sending message transaction, this process can be ignored because they do not involve related transactions. However, for the sending reputation score transaction, it is necessary to associate the corresponding transaction that causes the reputation score change. For the receiving transaction, the corresponding sending message transaction and the sending reputation score transaction need to be associated. Therefore, after receiving the sending reputation score transaction and receiving transaction, it is necessary to verify whether the associated transaction has been added to the blockchain. If the associated transaction exists, the transaction verification is passed; otherwise, the transaction is discarded.

Transactions that have passed the above 5-step verification are added to the transaction candidate set. The transaction candidate set is used to temporarily store verified transactions and wait for consensus.

4.4. Consensus Scheme. Consensus algorithm is the core part of the blockchain, and it can guarantee the consensus of distributed blockchain nodes. Common consensus algorithms currently include Proof of Work (PoW) used in Bitcoin system, Proof of Stake (PoS) used in Ethereum, and Practical Byzantine Fault Tolerance (PBFT) used in Hyperledger. By comparing PoW, PoS, and PBFT, Ayaz et al. [32] believe that PBFT has the most potential to become the consensus algorithms in the VANETs environment. The PoW consensus algorithm is discriminatory in computational power, and the POS consensus algorithm is discriminatory in stakes owned. However, the PBFT consensus algorithm is based on voting, which is not discriminatory in computational power and stakes owned. It can be seen that it is suitable for use in the VANETs environment. Therefore, PBFT-based consensus algorithm is adopted in this architecture.

The PBFT consensus algorithm has relatively high communication complexity, and its complexity is $O(n^2)$, where n is the number of nodes participating in the consensus. As you can see, performance degrades dramatically when there are many nodes in the network. Moreover, in PBFT consensus algorithm, if the byzantine nodes exceed 1/3 of the total number of participating consensus nodes, the consensus algorithm will fail. Based on these two considerations, this architecture selects nodes within M hop around the transaction producer rather than all nodes in the network, which can improve consensus efficiency. As a parameter in the consensus process, M is dynamically

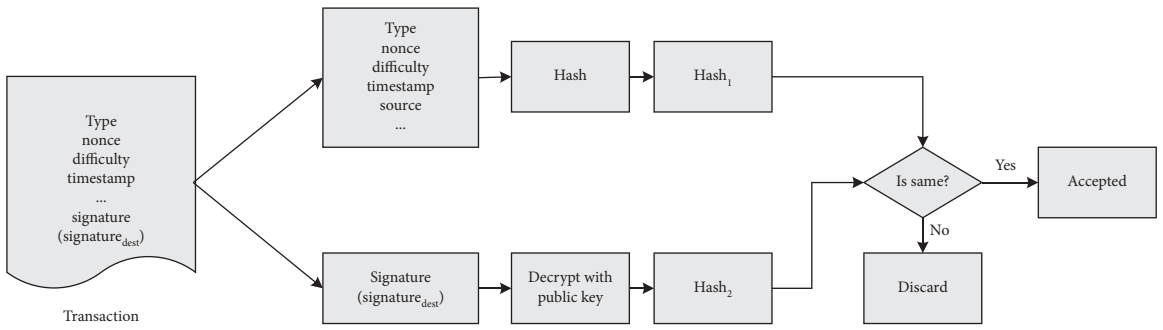

Figure 5: The signature verification process of the transaction.

adjusted according to the network environment. At the same time, node with reputation score greater than a certain threshold T is selected to participate in the consensus, so as to avoid the byzantine nodes participating in the consensus from exceeding 1/3 of the total number of nodes. If a node participating in the consensus is attacked and behaves maliciously, the RSU can quickly find and reduce its reputation score, thus preventing the attacked node from participating in the consensus process.

In this architecture, transaction consensus can be asynchronous, that is, transactions of different accounts, or transactions related to messages and reputation scores of the same account can be asynchronously agreed. Therefore, in the consensus process, all nodes can become the master node and present a consensus on transaction. But for a certain transaction, only one node can become the master node to avoid the occurrence of forks. However, in the traditional PBFT consensus algorithm, the selection method of the master node has the problem of forecasting in advance. In order to avoid a malicious attack on a certain transaction, the master node proposing a certain transaction should not be predicted in advance.

The verifiable random function VRF [33] maps the input value to a verifiable output. It is characterized by not being able to predict the random result and verifying the correctness of the result. In Algorand [34] and Ouroboros Praos [35], the authors use VRF to select the node proposing block, which has high security. Therefore, VRF is used to determine the master node of the consensus in this paper and PBFT consensus algorithm is improved. To achieve asynchronous consensus, this paper appends transaction information (the account‖source field and the timestamp field) to the VRF verification information. Transaction can be uniquely identified by additional transaction information. The detailed procedures are as follows.

(1) Each node G_i of scheduled generation block selects a transaction from the local transaction candidate set, uses the timestamp of transaction and the private key $G_i^{S_r}$ of node G_i as the input of VRF to generate the hash output of hash_value and proof_value through the VRF calculation function VRF_HASH and VRF proof function VRF_Proof. The calculation method of hash_value and proof_value is shown in the following:

$$\text{hash\_value} = \text{VRF\_HASH}\left(G_i^{S_r}, \text{timestamp}\right), \quad (2)$$

$$\text{proof\_value} = \text{VRF\_Proof}\left(G_i^{S_r}, \text{timestamp}\right). \quad (3)$$

(2) The node G_i sends the VRF verification message {hash_value, proof_value, account‖source, timestamp, account$_{G_i}$} to the surrounding nodes in the preprepare phase of the PBFT consensus algorithm, where the account$_{G_i}$ is the account of the node G_i. In particular, it should be noted that the nodes participating in the consensus are both the master node and the client node in this architecture. Therefore, the request phase is not involved in this architecture.

(3) After the surrounding nodes as VRF verifier receiving {hash$_{value}$, proof$_{value}$, account‖source, timestamp, account$_{G_i}$}, firstly they need to verify whether hash _value = VRF_$P2H$(proof_value) is true. If true, then zero-knowledge proof is performed through the VRF verification function VRF_Verify($G_i^{P_r}$, timestamp, proof_value), where $G_i^{P_r}$ is the public key of the node G_i. If the verification result is true, this means that the hash_value is generated by the node G_i through timestamp.

(4) Among the verified VRF verifiers for the same transaction {account‖source, timestamp}, only the node whose hash_value meets a certain condition is selected as the master node to perform other stages of the PBFT consensus, including prepare stage, commit stage, and reply stage. It should be noted that the client is not involved in this architecture, so the message is sent to the master node during the reply phase.

Through the above process, the master node can be randomly generated without being predicted in advance. At the same time, the asynchronous transaction consensus can be achieved.

As the vehicle moves, it may not be able to communicate with the RSU. In this case, the transaction that has reached consensus can only synchronize with the surrounding vehicles but cannot synchronize with the RSUs in real time. Therefore, it is necessary to use the Delay Tolerant Network (DTN). The vehicle firstly saves the transaction that has

reached consensus locally, and when it moves to the communication range of the RSU, it synchronizes the transaction to the RSU. In this way, all the transactions that have reached consensus can eventually be synchronized with the RSUs.

4.5. PoW Anti-Spam. Before nodes send a transaction, they first need to compute a random number satisfying a particular difficulty to complete the proof of work. Transactions cannot be sent until PoW is complete. When other nodes receive a transaction, they first need to verify whether nonce value is correct. Unlike PoW in Bitcoin, it is only used as an anti-spam tool in this paper. Since the amount of calculation is consumed during PoW, this can prevent malicious nodes from always sending malicious messages to occupy network bandwidth and perform DoS attacks.

In this paper, the difficulty of POW is related to the node's reputation score. The higher the node's reputation score, the lower the difficulty of POW. On the contrary, the lower the node's reputation score, the higher the difficulty of POW. The specific calculation method is shown in the following equations:

$$N_{\text{zero}} = \left\lfloor e^{-(\vartheta \cdot \text{Score}+\mu)} * N_{\text{hash}} \right\rfloor, \tag{4}$$

$$\text{difficulty} \Leftarrow \underbrace{0 \cdots 0}_{N_{\text{zero}}} + \left(2^{N_{\text{hash}} - N_{\text{zero}}} - 1\right)B, \tag{5}$$

where N_{hash} is the length of the hash value, which depends on the specific hash algorithm (e.g., the MD5 algorithm is 128 bits, the SHA-1 algorithm is 160 bits, and the SHA-256 algorithm is 256 bits). The specific selection of hash algorithms needs to be determined according to the application requirements. N_{zero} is the number of the first 0 in the POW difficulty string. The Score is the node's reputation score. ϑ and μ are two preset parameters, which are used to control the change rate and upper limit of POW difficulty. $\lfloor \cdot \rfloor$ indicates that the number is rounded down to the integer part. $(\cdot)B$ means convert a decimal number into a binary number.

As shown in Figure 6, POW is relatively difficult when the node's reputation score is very low. But with the increase of reputation score, the difficulty of PoW decreases gradually.

After the difficulty is determined, the PoW can be carried out for the transaction. The method is to try random numbers until a random number is found that satisfies the difficulty. The specific PoW method is shown in the following:

$$\text{Hash}(\text{type}, \text{difficulty}, \text{timestamp}, \text{nonce}) \leq \text{difficulty}. \tag{6}$$

4.6. Application Cases. In the VANETs environment, the communication between vehicles or between vehicles and RSUs is mainly used to improve traffic safety and efficiency and realize intelligent traffic management. Therefore, in the proposed architecture, application cases mainly include node creates accounts, node broadcasts messages, node obtains messages from other nodes, and RSU updates nodes' reputation score. Under this architecture, the working methods of these four application cases are as follows.

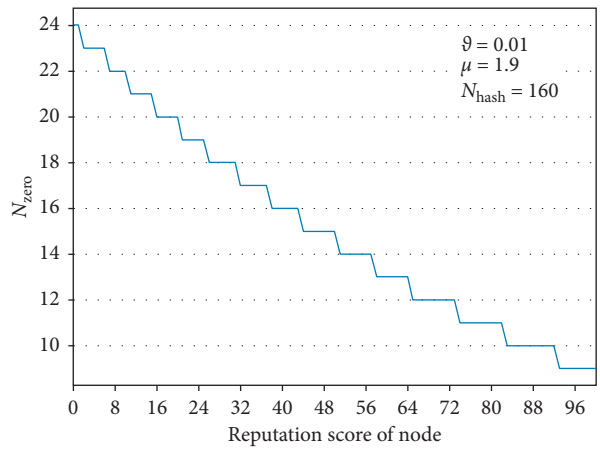

Figure 6: The relationship between PoW difficulty and reputation score.

4.6.1. The Node Creates Accounts. When a node joins the blockchain for the first time, an account needs to be created. The specific procedures are as follows:

1) Node $A \in \text{VNode}_i \| \text{RNode}_i$ registers information in the trusted center and obtains the public and private key pair $\{A^{P_r}, A^{S_r}\}$.
2) Node A generates a creating account transaction $T_{\text{acount\_create}}$, signs the transaction, and sends it to the RSU. The account field is the public key A^{P_r} of node A.
3) RSU $B \in \text{RNode}_i$ that receives $T_{\text{acount\_create}}$ verifies whether the public key account A^{P_r} has been registered in the TC.
4) If the verification is successful, $T_{\text{acount\_create}}$ will be verified, reached consensus, and added to the account chain of the account as the genesis transaction of the account.

4.6.2. The Node Broadcasts Messages. Vehicles or RSUs need to frequently broadcast some warning messages, road condition messages, and so on to improve traffic safety and efficiency. The specific procedures are as follows:

(1) Node $A \in \text{VNode}_i \| \text{RNode}_i$ generates a sending message transaction $T_{\text{message\_send}}$, signs it, and broadcasts it to surrounding nodes. It should be noted that the broadcast message does not have a definite destination node, and the transaction corresponding to the message needs to be stored in the node A's Message Chain. So, the destination field and the source field have the same content, that is, the account address of node A. The description information $\text{info}_{\text{desc}}$ of the message is recorded in the desc field of msg, and the content info_{pub} that is publicly accessible by other nodes is recorded in the

plaintext_cont field. The content $\text{info}_{\text{auth}}$ that needs to be authorized for other nodes to access is encrypted and recorded in the ciphertext_cont field, that is, ciphertext_cont = $E_{A^{P_r}}(\text{info}_{\text{auth}})$.

(2) After receiving $T_{\text{message\_send}}$ sent by node A, other nodes verify it, reach a consensus, and add it to the node A's Message Chain.

(3) After $T_{\text{message\_send}}$ is added to the blockchain, node A generates a receiving transaction T_{receive}, signs it, and sends it to the network.

(4) Once T_{receive} is received, other nodes verify the transaction, reach a consensus, and add it to the node A's Message Chain.

4.6.3. The Node Obtains Messages from Other Nodes. Node $B \in \text{VNode}_i \| \text{RNode}_i$ in network needs to obtain content from another node $A \in \text{VNode}_i \| \text{RNode}_i$ or obtain access to encrypted content in the Message Chain from node A. For node A, decrypting the ciphertext on the blockchain every time, encrypting it with the public key of node B, and then sending it to node B would undoubtedly cause a lot of trouble for node A as the data owner. Therefore, using proxy reencryption technology [36] in this paper adopts, node A only needs to encrypt the ciphertext with the reencryption key and send it to node B. The specific procedures for node B to obtain a message from node A are as follows:

(1) Node B generates a sending message transaction $T_{\text{message\_send}}$, signs it, and broadcasts to the surrounding nodes. The destination field is the account address of node A, and the desc field in the msg field records the content name that will be obtained from node A or the hash value of a block in the node A's Message Chain.

(2) After receiving $T_{\text{message\_send}}$ sent by node B, other nodes verify it, reach a consensus, and add it to the node B's Message Chain.

(3) Node A generates a receiving transaction T_{receive}, signs it, and sends it to the network. At the same time, node A generates ciphertext $\text{Data}_{\text{Encry}}$ based on what node B needs to obtain. The procedures of generating $\text{Data}_{\text{Encry}}$ are as follows:

(a) Node A generates a reencryption key $\text{Re} - \text{Key}_{A \longrightarrow B}$ based on its own public key A^{P_r} and public key B^{S_r} of node B.

(b) Node A generates the data Data_s sent to node B. If the content that node B needs to obtain is common content C_{comm} which is not on the blockchain, then C_{comm} needs to be encrypted with its own public key; that is, $\text{Data}_s = E_{A^{P_r}}(C_{\text{comm}})$. If the content that Node B needs to obtain is content M_{encry} on the blockchain, then $\text{Data}_s = M_{\text{encry}}$.

(c) Node A encrypts the data Data_s sent to node B by the reencryption algorithm; that is, $\text{Data}_{\text{Encry}} = \text{Re} - \text{Enc}(\text{Data}_s, \text{Re} - \text{Key}_{A \longrightarrow B})$.

(4) Node A generates a sending message transaction $T_{\text{message\_send}}$, signs it, and broadcasts it to surrounding nodes. The destination field is the account address of node B, and the ciphertext $\text{Data}_{\text{Encry}}$ of the message that needs to be sent to node B is recorded in the ciphertext_cont field.

(5) After receiving $T_{\text{message\_send}}$ sent by node A, other nodes verify it, reach a consensus, and add it to the node A's Message Chain.

(6) Node B decrypts $\text{Data}_{\text{Encry}}$ with its own private key to obtain the required content and at the same time generates a receiving transaction T_{receive} and sends it to the network.

(7) After receiving T_{receive} sent by node B, other nodes verify it, reach a consensus, and add it to the node B's Message Chain.

4.6.4. The RSU Updates Nodes' Reputation Score. The RSU $A \in \text{RNode}_i$ in network can update the reputation score of node $B \in \text{VNode}_i \| \text{RNode}_i$ according to its behavior, so as to motivate or punish node B. The specific procedures for node A to update the reputation score of node B are as follows:

(1) Node A generates a sending reputation score transaction $T_{\text{reputation\_send}}$, signs it, and sends it to node B. The destination field is the account address of node B. The associate field is the hash value of the corresponding transaction that causes the reputation score to change. The desc field of the reputation field describes the method of updating the reputation score, and the score field records the updated reputation score of node B.

(2) When node B receives $T_{\text{reputation\_send}}$, it first verifies the transaction. After the verification is passed, node B records the current time in the $\text{timestamp}_{\text{dest}}$ field, signs the transaction again, and broadcasts it to surrounding nodes.

(3) After receiving $T_{\text{reputation\_send}}$ sent by node B, other nodes verify it, reach a consensus, and add it to the node B's Reputation Chain.

(4) After $T_{\text{reputation\_send}}$ is added to the blockchain, node B generates a receiving transaction T_{receive}, signs it, and sends it to the network.

(5) After receiving T_{receive} sent by node B, other nodes verify it, reach a consensus, and add it to the node B's Reputation Chain.

During the reputation score update process, the RSU and the updated node are required to sign transaction simultaneously. The RSU signs the sending reputation score transaction and sends it to the corresponding vehicle. The corresponding vehicle signs again and then sends the transaction to the network to be added to blockchain. However, if the node (especially a moving vehicle) leaves the communication range of the current RSU when its reputation is updated, the forwarding is done with the assistance of other RSUs; if the node completely leaves the network (in

an offline state), the RSU saves the relevant transaction temporarily until the node joins the network and then updates. If a node performs malicious behavior and does not sign the sending reputation score transaction, the node will be punished and pulled into the blacklist so that it can no longer generate new transactions.

5. Security Analysis

This section describes the malicious attack scenarios that may occur in architecture and analyzes how this architecture deals with these malicious attacks.

5.1. The Sybil Attack. A malicious node may create multiple accounts and launch sybil attack on the network. However, in this architecture, public key, and private key pairs of vehicle and RUS are issued by TC, which is confirmed when the account is created. So, the malicious node cannot create large numbers of accounts to carry out the sybil attack.

5.2. The Transaction Flooding Attack. A malicious node may send a large number of malicious transactions to the network, which affects other nodes' judgment on the correctness of messages in the network. In this architecture, the node needs to complete the PoW before sending a transaction. The difficulty of PoW is related to the node's reputation score. After a node sends a malicious transaction, it will be discovered by the RSU and its reputation score will be reduced. This makes it increasingly difficult for node to initiate malicious attack. The node will abandon the attack voluntarily until PoW consumes more computational resources than the benefits of launching an attack.

5.3. The Precomputed PoW Attack. The timestamp is generated automatically by the system and cannot be changed by the node that generates the transaction. However, malicious node can save transactions that have completed PoW locally without sending them to the surrounding nodes. After a period of waiting, transactions within this period are sent out simultaneously, and then the network or node is attacked. In this architecture, after the node receives a transaction, it first verifies the transaction including the verification of the timestamp. If a malicious node waits for a period of time before sending a transaction, the interval between the timestamp of the transaction and the current time exceeds the threshold, and the verification cannot be passed. As a result, the transaction is discarded and cannot generate a valid attack.

5.4. The Conspiracy Attack. Two or more malicious nodes in the network may join together to provide false voting information during consensus process to conduct a conspiracy attack. The PBFT consensus method is used in this architecture. Consensus results can be affected when 1/3 of the malicious nodes participating in the consensus conduct a conspiracy attack. However, in this architecture, nodes within the M hop range around the production transaction node and with a reputation score greater than a certain threshold are selected to participate in the consensus. Because this architecture uses reputation score to motivate and punish nodes. Malicious nodes usually have low reputation scores, thus greatly reducing the probability of conspiracy attacks.

6. Architecture Verification

This section verifies proposed architecture, including the security verification of using Petri nets and the feasibility verification of PoW anti-spam through experiments.

6.1. Security Verification Based on Petri Net. Petri net is a mathematical tool for researching system. It can be used to describe and analyze asynchronous and concurrent system, as well as design and optimize system. Colored Petri Net (CPN) is a kind of advanced Petri Net that combines Petri Net and programming language. A token represents an object with a set of attributes. In CPN, each color set represents a different resource in the system, and the color set can be of multiple data types. At the same time, CPN supports hierarchical modeling, allowing the model to be partitioned into manageable parts that can be reused. Therefore, this paper adopts a top-down method to establish a hierarchical CPN model for the proposed V-Lattice.

6.1.1. The CPN Model. This paper defines the CPN through nine-tuples, namely, CPN = $(P, T, A, \Sigma, V, C, G, E, I)$. The top layer CPN model is shown in Figure 7, and the transition Verify Transaction is substitution transition as shown in Figure 8. The detailed introduction of the nine-tuple is as follows:

P is set of places, which is identified by an ellipse in figure. The upper right corner of each place is the initial function I, and the lower right corner is the color set function C. There are 6 places in the top layer CPN model, and 8 places in the substitution transition Verify Transaction.

T is the set of transitions that are identified by a rectangle in figure, in which the double-line rectangles represent substitution transition. The guard function G is located in the upper left corner of each transition. It is not involved in this model. There are 5 transitions in the top layer CPN model and 5 transitions in the substitution transition Verify Transaction.

A is the set of arcs that are used to connect places and transitions. Each arc contains an arc expression function E.

Σ is the set of color sets, and V is the set of variables. In this model, the definition of set of color sets and set of variables are shown in Table 5.

As shown in Figure 7, in the top layer CPN model, the Generate transition generates transactions which need to be sent and adds them to the list of transactions. The Send transition broadcasts the next transaction to be sent to the

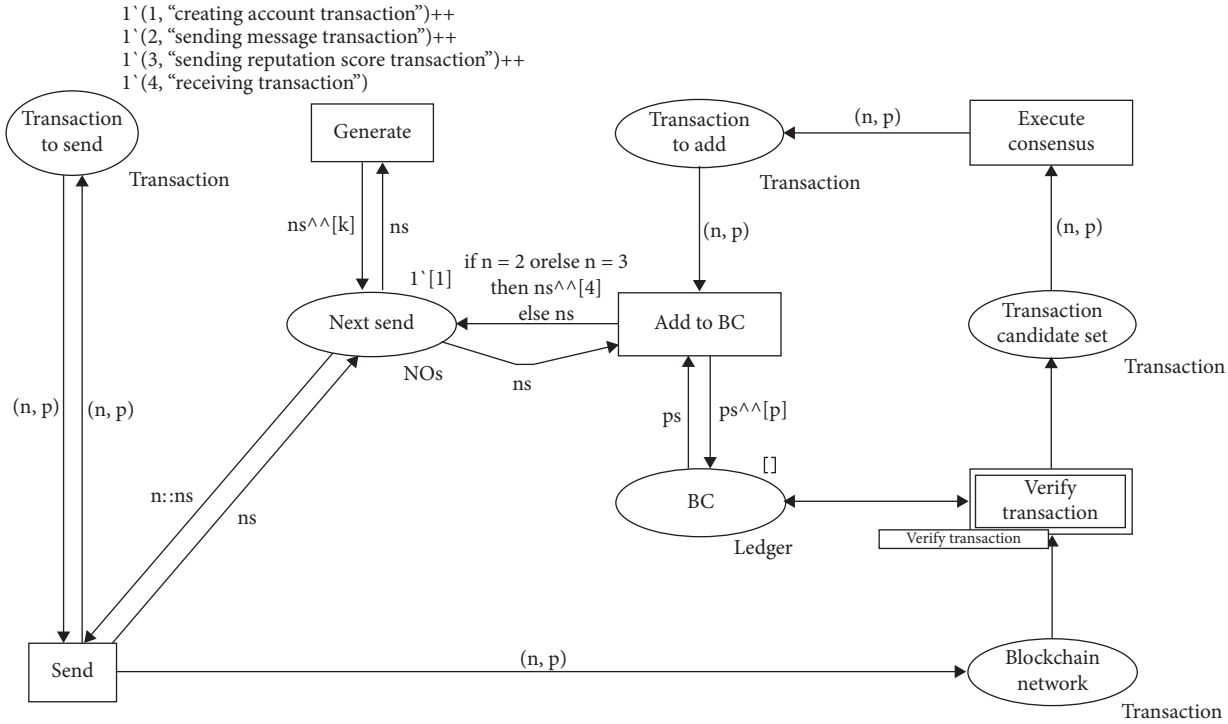

Figure 7: The top layer CPN model of V-Lattice.

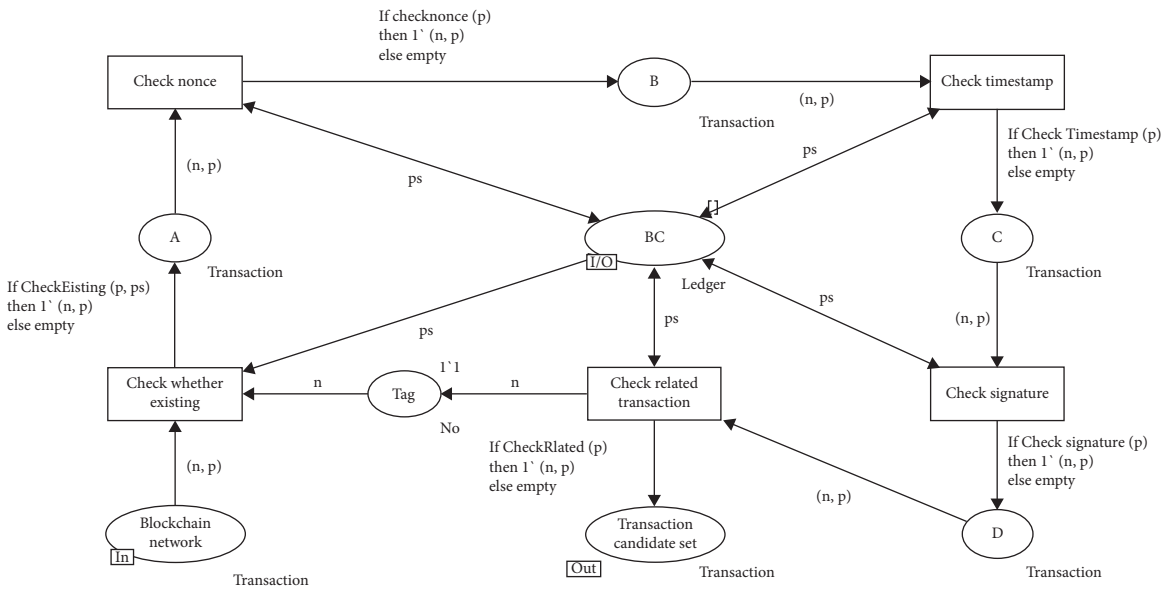

Figure 8: The subpage model of the Replacement Verify.

Table 5: The definition of set of color sets and set of variables.

Set of color sets	Description	Set of variables
Colset NO = int	Identify different transaction type	n
Colset NAME = string	Transaction name	p
Colset TRANSACTION = product NO ∗ NAME	Information transmitted	
Colset ledger = list NAME	Ledger in the blockchain	ps
Colset NOs = list NO	List of transactions to be sent	ns
Colset range = int with 1..3	Transaction type randomly generated	k

blockchain network. The Verify Transaction substitution transition is used to verify the transaction and add the verified transaction to the transaction candidate set. The Execute Consensus transition selects transactions from the transaction candidate set and reaches consensus. The Transaction to add transaction adds consensus-completed transactions to the blockchain. At the same time, for the sending message transaction and sending reputation score transaction, a receiving transaction is generated and added to the list of transactions that need to be sent.

In the subpage model of the Replacement Verify as shown in Figure 8, the Check whether existing transition checks whether transaction already exists in the blockchain. The Check nonce transition checks whether nonce value in the transaction is correct. The Check timestamp transition is used to check whether timestamp in the transaction is correct. The Check signature transition is used to check whether transaction is correctly signed by the sender of the transaction. The Check related transaction transition is used to check whether related transaction information is correct and has been added to the blockchain.

6.1.2. Security Analysis. The number of nodes in VANETs is finite, so the transactions generated by the Generate transition are also finite. There exists an integer M and the number of tokens in any place in the CPN model will not exceed M. Therefore, the CPN is bounded, that is, M-secure. The specific value of M depends on the speed with which transactions are generated and processed in the network. Modeling and simulating the network using CPN tools show that all transitions in the CPN are alive and can be fired indefinitely; that is, there is no dead transition. It can be seen that the workflow of architecture is secure. At the same time, through the analysis of CPN state space, it is found that every sending message transaction and sending reputation score transaction has a corresponding receiving transaction in the blockchain ledger, which ensures the existence of their pairing. It can be seen that the availability of architecture is secure.

In a word, using CPN to model architecture, it can be concluded that the architecture proposed in this paper is security.

6.2. Feasibility Verification of PoW Anti-Spam. In this section, the impact of different difficulty values on PoW time is analyzed by experiments, and the feasibility of PoW anti-spam is verified. The experiment in this paper is carried out in the environment of 2.8 GHz processor and 8 GB memory. At different difficulty values, 500 experiments were carried out independently, and then the average value was taken as the experimental result.

First, this paper verifies the impact of difficulty value on the times to find the nonce. As shown in Figure 9, the abscissa represents difficulty value (ranging from 1 to 24), and the ordinate represents the average times to find the nonce. It can be concluded from Figure 9 that the times for attempting to find a nonce increase exponentially as the difficulty value increases.

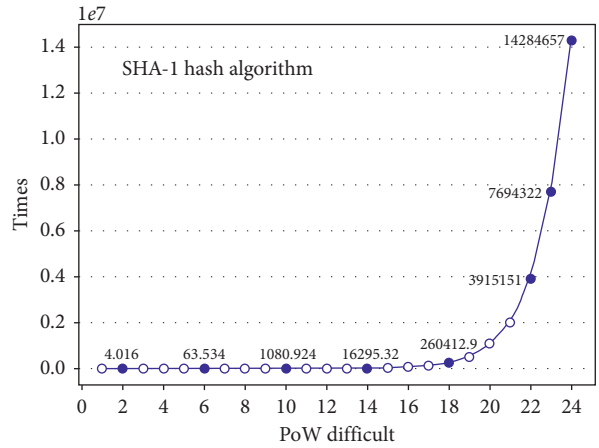

Figure 9: The relationship between difficulty and times to find the nonce.

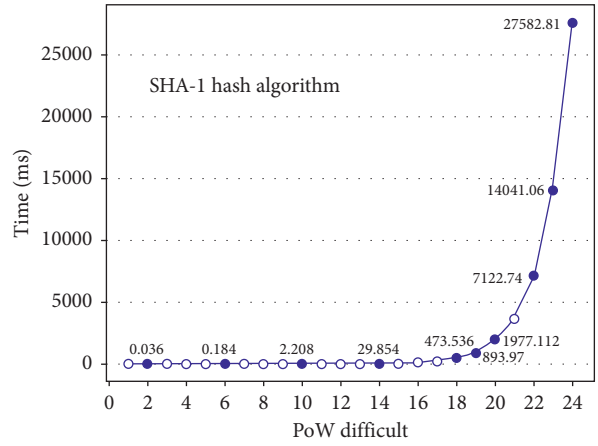

Figure 10: The relationship between difficulty and time to find the nonce.

This paper then verifies the impact of difficulty value on time required to find the nonce. As shown in Figure 10, abscissa represents difficulty value (ranging from 1 to 24), and ordinate represents average time required to find the nonce. It can be concluded from Figure 10 that time required to find a nonce increase exponentially as the difficulty value increases. When difficulty value is small, time required to complete the PoW is also short, basically in milliseconds level, which satisfies the needs of VANETs applications. When difficulty value is greater than or equal to 20, time required to complete the PoW reaches the second level. The messages sent by the nodes cannot satisfy the real-time requirements in the VANETs environment, especially for safety-related messages. However, for nodes with high difficulty values, their reputation scores are relatively low, and generated messages are also with low credibility. It can be seen that it is feasible to use PoW anti-spam in this architecture, and it can prevent nodes from performing malicious behaviors without affecting the performance of VANETs.

The average number of times to find nonce is not directly related to the hardware environment but is related to the selected hash algorithm. However, for the same hash algorithm, the average number of times is basically the same for all nodes. The main factor that affects time to find nonce is the computational power of the node. The more computational power, the less time required for the node to complete proof of work. Therefore, in order not to affect the performance of VANETs, it is necessary to dynamically adjust ϑ and μ shown in Section 4.5 according to the computational power of the nodes in the network.

7. Conclusions

This paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. In V-Lattice, each node (vehicle or roadside unit) has its own account chain, the transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. In order to encourage more nodes to participate in the blockchain, a reputation-based incentive mechanism is introduced in V-Lattice. At the same time, this paper proposes an asynchronous consensus scheme for transactions generated by nodes and describes how common applications (including node creates accounts, node broadcasts messages, node obtains messages from other nodes, and RSU updates nodes' reputation score) in the VANETs work under this architecture. This paper uses Colored Petri Nets to verify the security of the architecture and verifies the feasibility of PoW anti-spam through experiment. The validation results show that the architecture proposed in this paper is security and it is feasible to prevent nodes from generating malicious behaviors by using PoW anti-spam.

In future work, we will further study how to achieve efficient and asynchronous consensus on transactions under the architecture. At the same time, a blockchain pruning scheme will be designed to ensure that the blockchain can run normally on resource-constrained vehicles.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61862046, in part by the Inner Mongolia Autonomous Region Science and Technology Achievements Transformation Project under Grant CGZH2018124, and in part by the Science and Technology Program of Inner Mongolia Autonomous Region under Grant 2019GG376.

References

[1] S. Sharma, K. K. Ghanshala, and S. Mohan, "Blockchain-based internet of vehicles (IoV): an efficient secure ad hoc vehicular networking architecture," in *Proceedings of the 2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 452–457, IEEE, Dresden, Germany, September 2019.

[2] P. K. Sharma, S. Y. Moon, and J. H. Park, "Block-VN: a distributed blockchain based vehicular network architecture in smart city," *Journal of Information Processing Systems*, vol. 13, no. 1, pp. 184–195, 2017.

[3] M. Awais Hassan, U. Habiba, U. Ghani, and M. Shoaib, "A secure message-passing framework for inter-vehicular communication using blockchain," *International Journal of Distributed Sensor Networks*, vol. 15, no. 2, 2019.

[4] M. A. Rahman, M. M. Rashid, S. J. Barnes, and S. M. Abdullah, "A blockchain-based secure internet of vehicles management framework," in *Proceedings of the 2019 UK/China Emerging Technologies (UCET)*, pp. 1–4, Glasgow, UK, August 2019.

[5] X. 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.

[6] D. Zheng, C. Jing, R. Guo, S. Gao, and L. Wang, "A traceable blockchain-based access authentication system with privacy preservation in VANETs," *IEEE Access*, vol. 7, pp. 117716–117726, 2019.

[7] C. Wang, J. Shen, J.-F. Lai, and J. Liu, "B-TSCA: blockchain assisted trustworthiness scalable computation for V2I authentication in VANETs," *IEEE Transactions on Emerging Topics in Computing*, p. 1, 2020.

[8] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Network*, vol. 32, no. 6, pp. 184–192, 2018.

[9] L. Li, J. Liu, L. Cheng et al., "CreditCoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[10] H. Li, L. Pei, D. Liao, G. Sun, and D. Xu, "Blockchain meets VANET: an architecture for identity and location privacy protection in VANET," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1178–1193, 2019.

[11] Z. J. Lu, W. C. Liu, Q. Wang, G. Qu, and Z. L. Liu, "A privacy-preserving trust model based on blockchain for VANETs," *IEEE Access*, vol. 6, 2018.

[12] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.

[13] X. Liu, H. Huang, F. Xiao, and Z. Ma, "A blockchain-based trust management with conditional privacy-preserving announcement scheme for VANETs," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4101–4112, 2020.

[14] E. M. Cho and M. N. S. Perera, "Efficient certificate management in blockchain based internet of vehicles," in *Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 794–797, Melbourne, Victoria, Australia, May 2020.

[15] C. Lin, D. He, X. Huang, N. Kumar, and K.-K. R. Choo, "BCPPA: a blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2020.

[16] L. Zhang, M. Luo, J. Li et al., "Blockchain based secure data sharing system for Internet of vehicles: a position paper," *Vehicular Communications*, vol. 16, pp. 85–93, 2019.

[17] X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access*, vol. 7, pp. 58241–58254, 2019.

[18] S. K. Dwivedi, R. Amin, S. Vollala, and R. Chaudhry, "Blockchain-based secured event-information sharing protocol in internet of vehicles for smart cities," *Computers & Electrical Engineering*, vol. 86, 2020.

[19] IOTA: An Open, Feeless Data and Value Transfer Protocol, https://www.iota.org/.

[20] A. Churyumov, "Byteball: a decentralized system for storage and transfer of value," https://obyte.org/Byteball.pdf.

[21] "InterValue: connect, transfer and exchange all digital assets over the world," March 2018, https://www.inve.one/file/InterValue_whitepaper_cn.pdf.

[22] C. LeMahieu, "Nano: a feeless distributed cryptocurrency network," https://content.nano.org/whitepaper/Nano_Whitepaper_en.pdf.

[23] JURA: An Ultrafast, Feeless Self-regulated Decentralized Network, https://docsend.com/view/p9f87gp.

[24] B. Xu, T. Agbele, and R. Jiang, "Biometric blockchain: a secure solution for intelligent vehicle data sharing," in *Deep Biometrics*, pp. 245–256, Springer, Cham, Switzerland, 2020.

[25] A. S. Kulathunge and H. R. O. E. Dayarathna, "Communication framework for vehicular ad-hoc networks using Blockchain: case study of metro manila electric shuttle automation project," in *Proceedings of the 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pp. 85–90, Colombo, Sri Lanka, March 2019.

[26] Y. Yuan and F. Y. Wang, "Towards blockchain-based intelligent transportation systems," in *Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2663–2668, Rio de Janeiro, Brazil, November 2016.

[27] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "BlockChain: a distributed solution to automotive security and privacy," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.

[28] M. Singh and S. Kim, "Introduce reward–based intelligent vehicles communication using blockchain," in *Proceedings of the 2017 International SoC Design Conference (ISOCC)*, pp. 15-16, Seoul, Republic of Korea, November 2017.

[29] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*, pp. 137–140, New York, NY, USA, September 2016.

[30] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4640–4649, 2019.

[31] T. Zhou, X. Li, and H. Zhao, "DLattice: a permission-less blockchain based on DPoS-BA-DAG consensus for data tokenization," *IEEE Access*, vol. 7, pp. 39273–39287, 2019.

[32] F. Ayaz, Z. Sheng, D. Tian, G. Y. Liang, and V. Leung, "A voting blockchain based message dissemination in vehicular ad-hoc networks (VANETs)," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, June 2020.

[33] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 120–130, New York, NY, USA, October 1999.

[34] Y. Gilad, R. Hemo, S. M. Micali, G. Vlachos, and N. Zeldovich, "Algorand: scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*, pp. 51–68, New York, NY, USA, October 2017.

[35] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology – EUROCRYPT 2018*, vol. 10821, pp. 66–98, Springer, Cham, Switzerland, 2018.

[36] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.