

Received March 9, 2020, accepted April 28, 2020, date of publication May 4, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992203

# FADB: A Fine-Grained Access Control Scheme for VANET Data Based on Blockchain

HUI LI<sup>1,3</sup>, LISHUANG PEI<sup>1</sup>, DAN LIAO<sup>1,3</sup>, SONG CHEN<sup>2</sup>, MING ZHANG<sup>3,4</sup>, AND DU XU<sup>1</sup>

<sup>1</sup>School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

<sup>2</sup>30th Institute of China Electronic Technology Group Corporation, Haidian 610041, China

<sup>3</sup>Chengdu Research Institute, University of Electronic Science and Technology of China, Chengdu 611731, China

<sup>4</sup>Tianfu Collaborative Innovation Center, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Dan Liao (dliao.uestc@gmail.com)

This work was supported in part by the Project on Public Safety Risk Prevention and Control and Emergency Technical Equipment under Grant 2018YFC0831002, in part by the Sichuan Science and Technology Program under Grant 20ZDYF2832, in part by the National Natural Science Foundation of China under Grant 61971105, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2019J004 and Grant ZYGX2019J125, and in part by the Chengdu Science and Technology Program under Grant 2018YFYF00188GX.

**ABSTRACT** Vehicular Ad Hoc Network (VANET) is an important foundation of intelligent transportation system and is widely used in traffic management, automatic driving, and road optimization. With the gradual popularization and further development of VANET, a large amount of VANET data has been produced. However, it poses huge challenges to the security and privacy when using VANET data provides services for users. In this paper, combining the technologies of blockchain and ciphertext-based attribute encryption (CP-ABE), we propose a fine-grained access control scheme for VANET data based on blockchain (FADB). In FADB, we employ the blockchain to replace the third-party service providers for user identity management and data storage. And different VANET data access rights can be established according to user attribute. By improving the CP-ABE, the lightweight VANET devices can outsource complex encryption and decryption operations to powerful RSUs and further improve the efficiency of data access. Final, we carry out a series of simulation tests and security analysis, proving that the FADB can provide effective data security and low performance overhead.

**INDEX TERMS** VANET, blockchain, CP-ABE, access control.


## I. INTRODUCTION

With the rapid development of technologies such as mobile computing, wireless communication, and embedded devices, VANET is gradually maturing [1]. VANET is committed to improving transportation efficiency and driving safety. It can provide real-time data transmission and functional interaction for different vehicles and road facilities, allowing users to access Internet resources and share data at any time [1]. Nowadays, VANET, as an important part of the Internet of Things, has attracted widespread attentions in the academic and automotive industries [2].

In VANET, the vehicle is equipped with various sensors and communication equipment such as GPS, Wi-Fi, driving recorder and other value-added service equipment. By the communication of vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I), vehicles can exchange and share data

with each other, as shown in Figure 1. Therefore, VANET data, recorded through the equipment on vehicles, is the basis for a range of services offered by VANET. By using the VANET data, VANET can support diverse network services, such as more efficient traffic solutions, enhanced driving safety, and assisted driving. However, with the gradual development of VANET, more and more connected vehicles have joined. This has led to an exponential increase in the amount of VANET data. Faced with such a large amount of data, traditional local storage solutions are no longer suitable for VANET [3].

Therefore, a new storage method is needed in VANET to satisfy the large amount of data. On the basis of meeting the high concurrency and throughput of VANET data, the new storage method needs to effectively protect the security and privacy of data. Moreover, it can be able to honestly transfer and store data without third-parties. And we no longer worry about inaccessible data due to the third-party failures [4]. Fortunately, the emergence of blockchain technology has

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo .

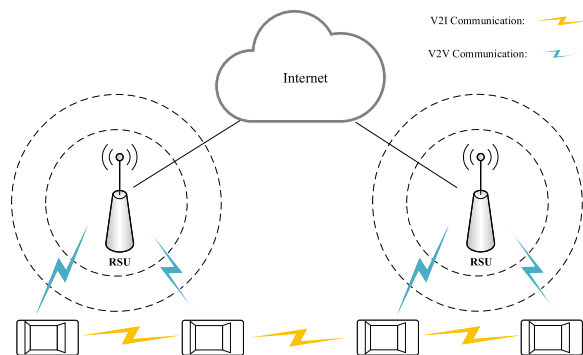


FIGURE 1. V2I&V2V communication.

made it possible to replace third-parties. Blockchain is a distributed database for data storage and retrieval [5]. It has strong non-destructive modification. In recent years, blockchain has been applied to different fields, such as financial service, resource sharing, trade management, and Internet of Things [6]. By using blockchain to store and share VANET data, the security of VANET data is effectively guaranteed [7]. This promotes the further circulation and sharing of VANET data, and generates greater value for VANET [8]. In this paper, by using blockchain technology, we implement the access control for VANET data. To protect the security and privacy of VANET data, the access control mechanism can prevent unauthorized entities from accessing data and ensure data confidentiality.

In addition, to meet the fine-grained access control for VANET data on cloud servers, the access control mechanism needs to implement data access control that can only be accessed and decrypted by particular users. However, traditional encryption schemes cannot satisfy the access control requirements. For example, the Advanced Encryption Standard (AES) [9] is difficult to send decryption key to the intended data access user. While the asymmetric encryption based RSA encryption algorithm requires data to encrypt file [10]. In RSA, the owner obtains the public key of each user before encrypting data. Any new user cannot access the data after encrypting data. This is a great limitation for multi-users access of data. Thus, driven by the need for fine-grained access control of data, the attribute based encryption (ABE) is proposed [11]. The ABE encryption scheme is designed for one-to-many encryption, maintaining fine-grained access control of data and data confidentiality. It is divided into two categories, the key policy attribute based encryption (KP-ABE) [12] and the ciphertext policy attribute based encryption (CP-ABE) [13]. The KP-ABE has an access policy attached to user key, while the CP-ABE has an access policy attached to ciphertext. CP-ABE is a data owner that can encrypt its data under a specified access policy through a set of attributes. It decrypts data when the attributes of data visitor meet the policy requirements. The CP-ABE does not require knowledge of who will access the data. It can provide greater flexibility and control during the encryption of data.

Thus, the CP-ABE is more suitable for storage solutions than KP-ABE.

In this article, we focus on the secure storage of VANET data and fine-grained access control. By utilizing the blockchain technology and CP-ABE algorithm, we propose a Fine-Grained Access Control Scheme for VANET Data based on Blockchain (FADB), which satisfies the distributed storage of VANET data. The contributions of this article are as follows:

- 1) We have proposed an access control scheme called FADB. By combining the CP-ABE encryption technology, Ethereum blockchain and IPFS [14], the FADB realizes distributed storage and fine-grained access for VANET data.
- 2) Based on the CP-ABE algorithm, we propose an enhanced security mechanism, called HECP-ABE. It combines blockchain technology to achieve step-by-step encryption and decryption operations, providing support for lightweight devices in VANET.
- 3) We test the credibility of our FADB through abundant simulation experiments. We have modified the Ethereum client to test the extra storage footprint and time spent by the entire data sharing process in the real deployment scenario.

The rest of the paper is organized as follows: In part II, we introduce the related work of this paper. The part III reviews the knowledge of CP-ABE encryption and blockchain. In part IV, we elaborate on the components of FADB and the interaction process between them. We elaborate the HECP-ABE algorithm and the detailed design of smart contract in part V. The part VI discusses the performance of FADB. Finally, we give the conclusion of this paper and the future research directions.

## II. RELATED WORK

### A. BLOCKCHAIN

Blockchain technology originated from the Bitcoin system [15]. As the economic value inherent in Bitcoin has gradually increased, more and more crypto currencies have emerged (Ethereum [16], Ripple [17], EOSIO [18]). This has greatly promoted the development of blockchain technology. At present, blockchain has been widely used in financial field [19] and some other fields. For example, in reference [20], blockchain was used to construct a new digital content distribution system, and the distributed copyright authentication mechanism was realized. The reference [21] constructed an anonymous data collection platform based on blockchain, which did not require a centralized trusted third party. In [22], the blockchain built a distributed shared platform for managing medical data.

A blockchain service framework for IoT data integrity was proposed in [23]. The framework provided more reliable data integrity verification for all data owners and users, and it did not rely on any trusted third-party auditing agency. In order to solve the security and privacy problems caused by

third-party storage of sensitive data, the authors [24] proposed a block-based data usage auditing architecture based on the layered identity-based encryption mechanism. It effectively protected users' privacy and ensured that data was shared confidentially with multiple service providers. In [25], a distributed cloud data architecture based on blockchain was proposed. This enabled tamper resistance to data sources, user privacy protection and reliable storage of data.

From the above researches, we can get that the blockchain-based distributed storage system can provide security and reliability far beyond the traditional storage method. It can avoid data loss and privacy leakage caused by third-party service providers. Therefore, by introducing the blockchain into VANET, we can effectively guarantee the secure storage of data and provide a durable and efficient data access service in this paper.

### B. CP-ABE

In [26], the Hierarchical Identity Based Encryption System (HIBE) and CP-ABE were combined to help enterprises effectively share confidential data on cloud servers. In order to solve the problem of some users' attributes affecting other users, an ABE scheme for avoiding user collusion was proposed in [27]. It effectively solved the attribute revocation problem by using the concept of attribute group. When the user revoked the attribute, the group manager would automatically update the keys of other users. In [28], a multi-authority cloud storage data access control scheme was proposed, which had efficient decryption and attribute revocation functions. It could achieve forward security and backward security. In [29], for the field of mobile cloud computing, an attribute-based encryption scheme for privacy-protected password policies was proposed. It enabled lightweight devices to outsource complex encryption and decryption operations to cloud service providers without revealing the content of the data. In [30], an efficient and revocable data access control scheme of multi-privileged cloud storage systems was designed as the basic technology of data access control scheme. In [31], it proposed an efficient revocable CP-ABE scheme for big data access control in cloud using proxy-based updates. The proxy server performed the ciphertext and secret key updates instead of data owner and data user respectively during revocation. In [32], the secure homomorphic encryption algorithm was combined with CP-ABE algorithm to construct a searchable CP-ABE access control scheme. The scheme not only ensured the security of data access, but also realized the retrieval of ciphertext and shortened query time.

### C. DATA SECURITY

In [23], it proposed a blockchain-based framework for data integrity service. Under such framework, a more reliable data integrity verification could be provided for both the data owners and data consumers without relying on any Third Party Auditor (TPA). This ensured the security of the data and provided a global authentication service. However,

the blockchain-based framework cannot provide users with fine-grained access control services. It restricts the efficiency and flexibility of data access.

In [27], it proposed an ABE scheme for cloud storage system with effective attribute revocation function to avoid user collusion ciphertext strategy. The problem of attribute revocation could be effectively solved by using the concept of attribute group. After revoking attributes from a user, the group administrator would update the keys of other users. However, this scheme does not adopt additional measures to ensure data security, which may cause some problems.

So, we combine the advantages of blockchain and CP-ABE. Through the blockchain to ensure data storage security and user identity authentication, use CP-ABE to achieve authorized access control of data to ensure efficient and flexible access. this paper designs HECP-ABE in FADB. It is possible to provide powerful distributed fine-grained data access services for VANET, which can greatly promote VANET data sharing.

## III. PRELIMINARIES

In this section, we mainly introduce the relevant background and preliminary knowledge designed in this paper. Tab. 1 shows some symbols and abbreviations involved in this article.

TABLE 1. Notations table.

SYMBOL	DESCRIPTION
$DO$	data owner
$DU$	data user
$PK$	system public key
$MSK$	system master key
$SK$	user secret key
$CC$	certification chain
$TC$	transaction chain
$\mathcal{L}$	system attributes set
$\mathcal{S}$	user attributes set
$RSU$	road side unit
$SC_{ic}$	smart contract of $IC$
$SC_{dc}$	smart contract of $DC$
$\mathbb{A}$	access structure
$F_{address}$	data file address
$kws$	keyword set
$CT_f$	encrypted file
$K$	decrypt key
$CT$	ciphertext
$CT'$	intermediate ciphertext

## A. ATTRIBUTE-BASED ENCRYPTION

### 1) BILINEAR MAP

Assume that there exist two cyclic multiplicative groups with big prime order,  $\mathbb{G}_0$  and  $\mathbb{G}_T$ . We set the  $g$  is a generator of  $\mathbb{G}_0$ . Then we can get a bilinear map  $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$  [13].

The bilinear map has the following properties:

- **Bilinear:**  $\forall u, v \in \mathbb{G}_0$  and  $a, b \in \mathbb{Z}_p$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$ .
- **Non-degeneracy:** there exist  $g_1, g_2 \in \mathbb{G}_0$ , making  $e(g_1, g_2) \neq 1$ .

- **Computable:**  $\forall u, v \in \mathbb{G}_0$ , there is an efficient computation  $e(u, v)$ .

2) ACCESS STRUCTURE

Let  $\mathcal{L} = \{a_1, a_2, \dots, a_n\}$  be the set of all attributes in the system. Then we call  $\mathcal{L}$  is the system attribute set. The  $a_i$  represents an attribute in the system. In this paper, the user attribute set is represented by  $\mathcal{S}$ , ( $\mathcal{S}$  should be a non-empty subset of  $\mathcal{L}$ ). Thus, we can construct the access structure  $\mathbb{A} \subseteq 2^{\{a_1, a_2, \dots, a_n\}}$  [13]. The characteristic of  $\mathbb{A}$  is monotone. If  $\forall B, C : B \in \mathbb{A}$  and  $B \subseteq C$ , we can get  $C \in \mathbb{A}$ .

3) ACCESS TREE

Let  $\mathcal{T}$  represent the access tree for an access structure  $\mathbb{A}$ , and  $x$  be a node of  $\mathcal{T}$ . Thus,  $\mathcal{T}_x$  is represented the sub-tree of  $\mathcal{T}$  rooted at the node of  $x$ . We set the  $R$  as the root node of  $\mathcal{T}$ . Then, if  $x = R$ ,  $\mathcal{T}_x$  can be seen as  $\mathcal{T}_R$ . For each non-leaf node of  $\mathcal{T}_x$ , it is described by its children node number  $num_x$  and threshold gate  $k_x$ , where  $k_x \in [1, num_x]$ . When  $k_x = 1$ , the threshold gate  $k_x$  is an OR gate. When  $k_x = num_x$ , the threshold gate  $k_x$  is an AND gate. For each leaf node of  $\mathcal{T}_x$ , it is stated by user attribute set  $\mathcal{S}$  and threshold  $k_x$ . If user attribute set  $\mathcal{S}$  satisfies  $\mathcal{T}_x$ , we donate  $\mathcal{T}_x(\mathcal{S}) = 1$ .  $\mathcal{T}_x(\mathcal{S})$  can be computed recursively. If  $x$  is a non-leaf node, we compute all children nodes denoted by  $x'$ . If the number of  $\mathcal{T}_{x'}(\mathcal{S})$  is more than  $k_x$ ,  $\mathcal{T}_x(\mathcal{S})$  outputs 1. If  $x$  is a leaf node, and  $att(x) \in \mathcal{S}$  ( $att(x)$  denotes the attribute related to the leaf node  $x$ ),  $\mathcal{T}_x(\mathcal{S})$  outputs 1.

In this paper, we define some functions to work with the access tree:  $parent(x)$  denotes the parent node of  $x$  in  $\mathcal{T}$ ;  $att(x)$  denotes the related attributes of  $x$ ;  $num(x)$  denotes the number of children nodes of  $x$ ;  $index(x)$  denotes the index of each children node of  $x$ .

**B. BLOCKCHAIN TECHNOLOGY**

1) BLOCKCHAIN

The blockchain originated from a paper entitled ‘‘Bitcoin: A Peer-to-Peer Electro-nic Cash System’’, which was published by a scholar named Nakamoto Satoshi [15]. It proposed a de-trusted cryptocurrency called Bitcoin, which was officially released and had been running smoothly for ten years. The success of Bitcoin has led to the rapid prosperity of cryptocurrencies, and further contributes to the rapid development of the underlying blockchain technology. Currently, blockchain technology has been widely applied in the field of financial services, and gradually expanded to asset registration, social credit investigation, resource sharing and other industries. Blockchain technology has the following characteristics:

- **Decentralized control:** All transactions in blockchain are voted by all nodes, which avoids the centralization of the blockchain.
- **Unalterable:** Each node stores the same copy, and the copies can be verified between nodes. This makes it

necessary to change 51% of copies to change the records of the entire system.

- **Irreversibility:** Transactions recorded on the blockchain cannot be deleted or changed after a certain number of confirmations.

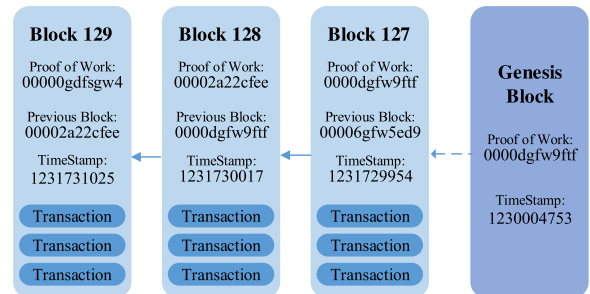


FIGURE 2. Blockchain struct.

Blockchain is a distributed database that backs up all transactions to each node. It consists of a series of block links. As shown in Figure 2, each block contains a hunk and a series of transactions. The hunk contains the timestamp, version number, hash of the previous block, and transaction information for the Merkle root tree. The old blocks are locked when a new block is created by referencing the hash value of the previous block. By creating new blocks in this way, a growing chain structure is formed. Block generation is the consensus result of the whole blockchain network. This mechanism guarantees the inelastic modification of blockchain.

2) ETHEREUM

Ethereum is seen as a blockchain 2.0 platform that supports reliable cryptocurrency transactions and smart contracts [33]. Compared to Bitcoin, Ethereum has a built-in Turing-complete programming language. This allows users to programmatically create, compile, deploy, and run a variety of standardized, scalable, and fully featured smart contracts. Once the smart contract is deployed, the smart contract can be invoked to complete the corresponding transaction.

a: ETHEREUM ACCOUNT

In Ethereum system, the user state consists three parts: account object state, the transfer state of value, and the state of information transition. In general, Ethereum has two types of accounts: external accounts (EOAs, controlled by private key) and contract accounts (controlled by contract code). The EOAs is controlled by user through a personal private key, which is sent by creating and signing a transaction. When the contract account receives messages, it activates the internal code and reads/writes the internal storage.

b: SMART CONTRACT

A smart contract is a computer trading agreement that runs on the terms of an implementation contract in Ethereum [34]. It is located at a specific location in Ethereum blockchain and is stored in Ethereum specific binary format (EVM bytecode).

The smart contract is called by the Ethereum Virtual Machine (EVM). Once a smart contract is deployed to the EVM, it can be automated and self-verified without manual intervention. It interacts with smart contracts through contract addresses and application interfaces. Smart contracts guarantee the stability and efficient operation of the system in the environment where there is no trusted third party.

Therefore, by introducing Ethereum into VANET, an equally decentralized trust system is constructed. The problems of centralization in the original data sharing scheme are solved, such as the data loss caused by third party, privacy disclosure, key abuse and other security risks.

### 3) IPFS

Inter Planetary File System (IPFS) is a global, peer-to-peer distributed version of the file system. The goal of IPFS is to connect all computing devices with a unified file system. IPFS can be seen as a complement and improvement to the Hypertext Transfer Protocol (HTTP). But it also act as a standalone BitTorrent cluster. By combining distributed hash tables (DHT), block exchange incentives, and self-certified namespaces, IPFS has no single point of failure and no mutual trust between nodes. In addition, IPFS provides a high-throughput content-addressable block storage model with content-addressed hyperlinks. After the file is uploaded to IPFS, it generates a hash string, which is used to retrieve the file. IPFS implements content-based addressing in this way. File distribution uses a BitTorrent-based protocol that enables files to be transferred, stored, and accessed in a distributed manner. This can conducive to saving bandwidth and preventing the DDoS attacks of HTTP protocol.

IPFS is a new generation of distributed data storage solution. It supports large-scale persistent storage of data and provides version control to facilitate management of data at different stages. Therefore, applying the IPS to VANET can effectively improve the security of data and provide support for large-scale data sharing in VANET.

## IV. SYSTEM MODEL

The proposed FADB describes a new type of distributed VANET data storage and access control system by combining Ethereum, CP-ABE, IPFS and other technologies. Figure 3 describes FADB's system model in detail including the system components and interactions.

### A. SYSTEM COMPONENT

#### 1) DATA OWNER (DO)

DO is the data producer. It is a group of vehicles or devices that need to share data in VANET. Due to its performance limitations, DO does not have high computing performance and large data storage space. Thus, DO does not have the conditions for large-scale storage and sharing of data in this paper. DO can upload data to the IPFS through the RSU. Due to the performance limitations of DO and its own dynamic nature, the blockchain network in FADB does not

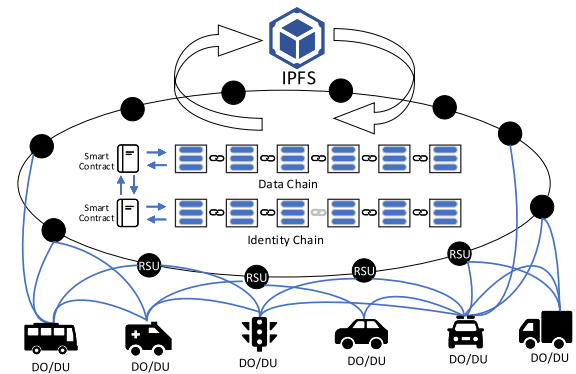


FIGURE 3. System model.

contain DO. In the blockchain network, we use RSU as proxy node to process transactions that related to data upload.

#### 2) DATA USER (DU)

DU is the data consumer. And DU needs to request data through the proxy RSU in VANET. Sometimes, DO and DU may be the same entity at the physical level. For example, a vehicle may request and share data at the same time. In this paper, both DOs and DUs are the users in our system model.

#### 3) RSU

RSU is a communication unit distributed along a certain distance on both sides of road, and each RSU is equipped with an improved Ethereum client, sufficient processing performance, storage space and good network connection. In FADB, we build the entire blockchain network with RSUs as the nodes. The RSU proxies the data upload and access operations of DOs and DUs within its coverage, while leveraging its high performance to perform most of the work in data encryption and decryption. This alleviates the performance requirements of DOs and DUs, enabling data sharing more efficient.

#### 4) IDENTITY CHAIN (IC)

In FADB, IC manages user's identity registration and changes. Each transaction stored in IC corresponds to user's identity information. Transactions are encrypted by smart contracts. Only through smart contracts can they access user's identity information. This effectively prevents user's privacy from being leaked and misused. Each user is registered on IC when it enters FADB system. The registration method is to authenticate user through RSU. After the authentication is passed, the RSU calls the smart contract belonging to IC to generate a private key for user. The smart contract constructs a transaction and records it on IC, the transaction containing the authenticated identity information of user.

The main fields in the transaction of IC are the following:

- Device ID: This field is user's unique identifier. When user makes a property change, Device ID is used to confirm whether it is the same user.

- Attribute set: This field is stored in  $\mathcal{S}$ . When user's attributes are changed, the type of data that the user can access in FADB also changed.
- Private key: This field stores user's private key. The ability to access data through a private key is a representation of user rights.

### 5) SMART CONTRACT OF IDENTITY CHAIN ( $SC_{ic}$ )

A set of operating methods for  $IC$  are defined in  $SC_{ic}$ .  $SC_{ic}$  is responsible for the generation of  $MSK$  and  $PK$ . After the  $SC_{ic}$  completes the generation of  $MSK$ ,  $RSU$  can write user's registration information to  $IC$  by calling  $SC_{ic}$ . When  $DU$  accesses the data, the  $SC_{ic}$  needs to perform preliminary verification to confirm whether the  $DU$  is a registered user in FADB.

### 6) DATA CHAIN ( $DC$ )

With respect to  $IC$ ,  $DC$  is another blockchain in FADB. The data stored in  $DC$  is a metadata that the user uploads to IPFS. The transaction in  $DC$  includes the following fields:

- *hash*: This field is used to ensure the correctness of data. It is to avoid problems such as incomplete files caused by network transmission errors.
- *kws*: This field means the keyword digest of data. It is represented by *kws*. In this paper, the *kws* is public, and is used to match data by matching the keyword.
- *CT*: The decryption key of data is encrypted and stored in this field. This field *CT* can only be accessed by smart contract.
- *F<sub>address</sub>*: This field means the address of data file in IPFS. It is represented by *F<sub>address</sub>*. By using *F<sub>address</sub>*, the encrypted data file stored in IPFS can be accessed.

### 7) SMART CONTRACT OF DATA CHAIN ( $SC_{dc}$ )

$SC_{dc}$  has the  $DC$  operating rights for reading, writing, and retrieving. When data is uploaded to IPFS, a transaction is generated by  $SC_{dc}$  and the transaction is written to  $DC$ . The flag of successful data upload is that the data can be retrieved and accessed by  $DC$ . First,  $DU$  calls the  $SC_{dc}$  through  $RSU$ . Then, the  $SC_{dc}$  calls the  $SC_{ic}$  to verify the  $DU$  identity on  $IC$ . Final,  $SC_{dc}$  sends  $DC$  to  $RSU$  after the verification is passed.

## B. SYSTEM INTERACTION

In FADB, the interactions between components mainly include the following stages: user registration, data upload, and authorized access. Here, we introduce them separately.

### 1) SYSTEM INITIALIZATION

The FADB system is initialized, as shown in Figure 4. When  $SC_{ic}$  is deployed to FADB system, it executes the  $Setup(1^\lambda, \mathcal{L}) \rightarrow (PK, MSK)$  algorithm, and generates the  $MSK$  and  $PK$ . The input parameters of the  $Setup$  algorithm are the system security parameter  $\lambda$  and system attribute set  $\mathcal{L}$ .  $\lambda$  is a constant that guarantees the security of the system. And it specifies the length of key generation. After completing the

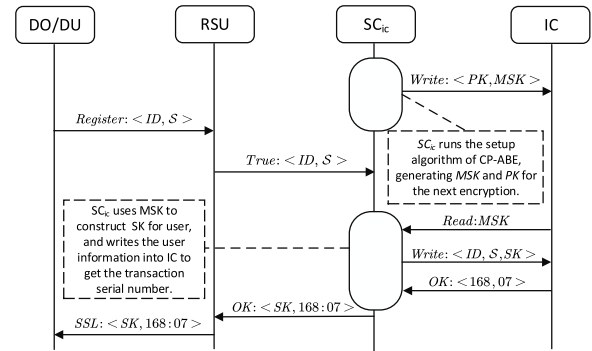


FIGURE 4. System initialization & User registration.

above steps,  $SC_{ic}$  constructs a transaction to write  $PK$  and  $MSK$  to the  $IC$ . The  $MSK$  is encrypted by  $SC_{ic}$  and only  $SC_{ic}$  can access it. The  $PK$  is cached in each  $RSU$ , ensuring that any node can access it. After system initialization, users can access the system by registering.

### 2) USER REGISTRATION

When user enjoys our FADB system, he/she sends a registration request to  $RSU$ . In FADB system, user connects to the  $RSU$  through a wireless access technology based on the IEEE 802.11p protocol. The registration request contains the device ID and user attribute set  $\mathcal{S}$ . The  $RSU$  verifies the authenticity and validity of registration message. After verification is passed, the  $RSU$  forwards the registration message to the  $SC_{ic}$ . The user registration algorithm,  $Register(MSK, \mathcal{S}) \rightarrow SK$ , is executed to generate the  $SK$  by registration algorithm. The  $SK$  is bound to  $\mathcal{S}$ . When users accesses the data,  $SK$  can be used to verify whether users satisfies the access requirements. After the transaction is constructed,  $SC_{ic}$  writes the transaction to  $IC$  and finally gets the serial number  $UID : < XX, YY >$  of the transaction in  $IC$ . For example, in Figure 4,  $OK < SK, 168 : 07 >$  means  $OK < SK, XX : YY >$ .  $XX$  represents the block number of the transaction, and  $YY$  is the transaction number in the block. Through the serial number,  $XX$  can directly locate the block where the transaction is located, and  $YY$  can directly determine where the transaction is stored in the block. This reduces the complexity of the query to a constant level. Thus, we can quickly retrieve user's identity information on the chain by  $XX$  and  $YY$ . Finally,  $SC_{ic}$  returns  $UID$  and  $SK$  to  $RSU$ . After receiving the message of successful registration,  $RSU$  sends  $UID$  and  $SK$  to the user, and the registration is completed.

### 3) DATA UPLOAD

In this stage,  $DO$  uses  $RSU$  as a proxy to upload data to IPFS, as shown in Figure 5. A short-range and high-speed connection is established between  $DU$  and  $RSU$  through the IEEE802.11P protocol. Before data is uploaded, the file  $F$  is encrypted by AES. The input of the encryption algorithm  $FileEncrypt(F) \rightarrow (CT_f, K, kws)$  is  $F$ , and the output is the encrypted file  $CT_f$ , file decryption key  $K$ , and the file

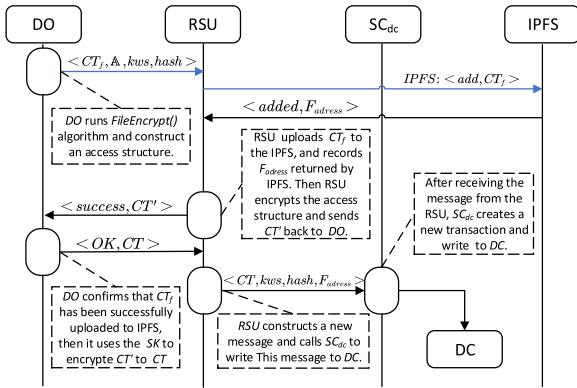


FIGURE 5. Data upload.

retrieval keyword set  $kws$ . After the file  $F$  is encrypted,  $DO$  constructs the access structure  $\mathbb{A}$ . It can access file  $F$  when user attributes set  $\mathcal{S}$  satisfies the access policy defined in  $\mathbb{A}$ . Then, the  $DO$  constructs a data upload request message and sends it to  $RSU$ .  $RSU$  establishes a stable data connection channel with  $DO$  to receive  $CT_f$ . Then  $RSU$  executes the encryption algorithm  $RSU.Encrypt(PK, \mathbb{A}) \rightarrow CT'$ . It alleviates the  $DO$  performance requirements of running the complete HECP-ABE algorithm. Because we have moved most of the encryption operations to  $RSU$ . After the  $RSU$  completes the encryption, it generates an intermediate ciphertext  $CT'$  and sends a successfully uploaded message back to  $DO$ . After  $DO$  confirms the message, it generates a ciphertext  $CT$  for the  $CT'$  by running the algorithm  $DO.Encrypt(PK, CK, CT') \rightarrow CT$ . Then  $DO$  sends  $CT$  to  $RSU$ . After receiving the  $CT$ , the  $RSU$  constructs a new message and sends to  $DC_{dc}$ . This message contains  $kws$ ,  $hash$  and  $F_{address}$ . After  $SC_{dc}$  receives the message, it constructs a transaction based on the message and finally writes the transaction to  $DC$ . When the transaction is successfully written into the blockchain, it indicates that the VANET data has been successfully uploaded and can be retrieved by all  $DUs$ .

the keyword from the  $DC$ . The other is to call the  $SC_{ic}$  to verify the  $UID$  of  $DU$ , and get  $DU$ 's partial private key for decryption in  $RSU$ . After retrieving the eligible transaction,  $SC_{dc}$  reads  $F_{address}$ ,  $CT$  and  $hash$  from the transaction.

When  $SC_{ic}$  received the request from  $RSU$ , it obtains the corresponding user identity information from  $IC$  according to the  $UID$ . If the  $UID$  is forged or unregistered by  $DU$ , the  $RSU$  does not get the correct  $CT'$ . Thus, it guarantees the security of data. If the  $UID$  is correct, the  $DU$  identity information can be quickly retrieved based on  $UID$ . The  $SK$  is intercepted to construct a key  $SK'$  for  $RSU$  decryption. After the  $SC_{dc}$  successfully receives replies from both  $DC$  and  $SC_{ic}$ , it constructs a new message  $\langle hash, F_{address}, SK', CT \rangle$  and sends to  $RSU$ . Then the  $RSU$  requests  $CT_f$  from IPFS based on  $F_{address}$  in the message  $\langle hash, F_{address}, SK', CT \rangle$ .

When  $RSU$  obtains  $CT_f$ , it checks the correctness of  $CT_f$  according to the hash. If  $CT_f$  is corrupted due to transmission, the  $RSU$  needs to request  $CT_f$  again from IPFS. After that, the  $RSU$  runs the decryption algorithm  $RSU.Decrypt(PK, CT, SK') \rightarrow CT'$ , and the  $CT$  is initially decrypted by  $PK$  and  $SK'$ . If the  $DU$  does not satisfy the decryption request of  $CT$ , the  $RSU$  fails to decrypt. The access failure message is sent back to  $DO$ .  $CT'$  is generated when the  $RSU$  decrypts  $CT$  successfully. Then, the  $RSU$  sends  $CT'$  and  $CT_f$  to  $DO$ . The  $DO$  executes the decryption algorithm  $DO.Decrypt(CT', SK) \rightarrow K$ , which decrypts the  $CT'$  based on the original  $RSU$  decryption to obtain the encrypted content. The  $DO$  only needs one calculation to get  $K$ , making the decryption speed much higher. Then,  $DO$  needs to use  $K$  to perform AES decryption on  $CT_f$  to access data. After the above steps,  $DO$  completes fine-grained access to the shared data.

## V. ALGORITHM DESIGN

The FADB is designed on the basis of Ethereum and CP-ABE. In the previous section, we have introduced the entire process of FADB's fine-grained access to data and how smart contracts guarantee data security and privacy.

Below we introduce the algorithms involved in FADB, HECP-ABE and smart contract. By improving the CP-ABE algorithm, we have designed the HECP-ABE algorithm so that it can ignore the difference in hardware performance in VANET and has a good performance improvement. In this section, we detail the implementation principle of HECP-ABE algorithm. In addition, we have improved the Ethereum client so that it can handle two parallel blockchains. Therefore, this section also gives a detailed introduction to the key points and pseudo-code implementation of smart contract.

### A. HECP-ABE ALGORITHM

HECP-ABE is a specific attribute encryption algorithm for VANET. It is improved on the basis of CP-ABE. It solves the CP-ABE' problem of demanding too much from VANET devices. In HECP-ABE, most of the encryption and decryption operations are transferred from users to the  $RSUs$ .

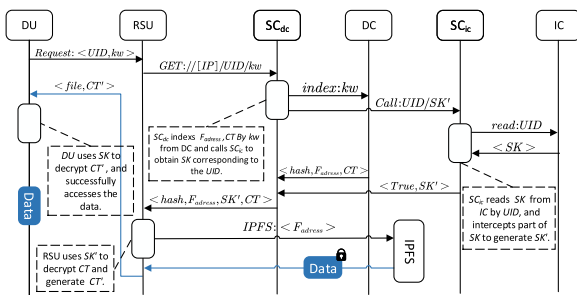


FIGURE 6. Authorized access.

### 4) AUTHORIZED ACCESS

In this stage,  $DU$  can access data through  $RSU$ . Figure 6 shows the specific process of authorized access. First, a  $DU$  sends a data request  $\langle UID, keyword \rangle$  to  $RSU$ .  $RSU$  invokes the  $SC_{dc}$  and sends this request.  $SC_{dc}$  splits the request into two parts. One is to retrieve the data by matching

Thus, the VANET devices can be lightweight without high computing and storage capacity.

*Phase 1 (Initialization):*  $Setup(1^\lambda, \mathcal{L}) \rightarrow (PK, MSK)$ . The setup algorithm chooses two bilinear group  $\mathbb{G}_0$  and  $\mathbb{G}_T$ . It randomly chooses three elements  $\alpha, \beta \in \mathbb{Z}_p$  and  $h \in \mathbb{G}_0$ . For each  $a_i \in \mathcal{L}$ , the algorithm chooses a random  $v_i$  and computes  $PK_i = g^{v_i}$ . Thus, the public key is published as:

$$PK = \{\mathbb{G}_0, g, h, g^\alpha, g^\beta, e(g, g)^{\alpha\beta}, \{PK_i = g^{v_i} | a_i \in \mathcal{L}\}\}$$

And the master key is  $MSK = \{\alpha, \beta\}$ . The setup algorithm is automatically executed by  $SC_{ic}$  when it is deployed to the blockchain network. The blockchain makes sure the proper generation and secure storage of  $MSK$  and  $PK$ .

*Phase 2 (User Register):*  $Register(MSK, \mathcal{S}) \rightarrow SK$ . The register algorithm takes  $\mathcal{S}$  and  $MSK$  as input. It outputs a secret key  $SK$ . In user register function of HECP-ABE algorithm, it first selects a random  $\gamma \in \mathbb{Z}_p$ , which is a unique assigned to each of users in FADB. After that, it random chooses  $\varepsilon \in \mathbb{Z}_p$  for each attribution in  $\mathcal{S}$ . Finally, it computes the user secret key as:

$$SK = \{D = g^{(\alpha+\gamma)\beta}, D_1 = g^\gamma h^\varepsilon, D_2 = g^\varepsilon, \forall a_i \in \mathcal{S} : D_i = g^{\beta\gamma v_i^{-1}}\}$$

*Register(MSK, S)* is executed by  $SC_{ic}$ . After the  $SC_{ic}$  receives a user registration request from  $RSU$ . This registration request contains the user's attribute set  $\mathcal{S}$ , the  $MSK$  is stored in  $IC$  and read by  $SC_{ic}$ . After finishing the register, the  $SC_{ic}$  return the user's private key  $SK$  through a secure channel.

*Phase3 (Encryption):*  $FileEncrypt(F) \rightarrow (CT_f, K, kws)$ . This encryption algorithm computes  $CT_f = Enc_{AES}(F)$  and sends  $CT_f$  to  $RSU$ , then keeps  $K$  for the next encryption algorithm in the iteration of HECP-ABE.

*RSU.Encrypt(PK, A) → CT'*. When the function works, it first selects a polynomial  $q_x$  for each node  $x$  in the access tree  $\mathcal{T}$ . The polynomial is beginning from the root node of  $\mathcal{T}$ . In a top-down manner, each node in  $\mathcal{T}$  is selected a polynomial. Here, The threshold value  $k_x$  is more than the degree  $d_x$  of  $q_x$ , like that,  $k_x = d_x + 1$ . Starting from the root node  $R$ ,  $RSU$  selects a random  $s_1 \in \mathbb{Z}_p$  and sets  $q_R(0) = s_1$ . After that, it chooses  $d_R$  other points to define  $q_R$  completely. For any other node  $x$  in  $\mathcal{T}$ , it sets  $q_R(0) = q_{parent(x)}(index(x))$ , and selects  $d_x$  other points randomly to completely define  $q_x$ .

Then, the intermediate ciphertext  $CT'$  can be constructed as:

$$CT' = \{\mathcal{T}, C'_3 = g^{\beta s_1}, C'_4 = h^{\beta s_1}\} \\ C_i = g^{v_i q_x(0)}, \quad \forall a_i = att(x) \in X$$

The  $X$  in  $CT'$  is the set of attributes associating with the leaf nodes.

*DO.Encrypt(PK, K, CT') → CT*. In this algorithm, the  $DO$  randomly chooses  $s \in \mathbb{Z}_p$  and computes that:

$$C_1 = K \times e(g, g)^{\alpha\beta s}, \quad C_2 = g^s, \\ C_3 = C'_3 \times g^{\beta s}, \quad C_4 = C'_4 \times h^{\beta s}$$

Finally, it outputs the ciphertext as:

$$CT = \left\{ \begin{array}{l} \mathcal{T}, E_K(M), C_1 = K \times e(g, g)^{\alpha\beta s}, \\ C_2 = g^s, C_3 = g^{\beta s_1} \times g^{\beta s}, C_4 = h^{\beta s_1} \times h^{\beta s}, \\ C_i = g^{v_i q_x(0)}, \forall a_i = att(x) \in X \end{array} \right\}$$

*Phase 4 (Decryption):*  $RSU.Decrypt(PK, CT, SK') \rightarrow CT'$ . The  $SK'$  is a part of user's secret key, and  $SK' = \{D_1 = g^\gamma h^\varepsilon, D_2 = g^\varepsilon, \forall a_i \in \mathcal{S} : D_i = g^{\beta\gamma v_i^{-1}}\}$ . The  $CT$  is got from the  $SC_{dc}$ . The  $RSU.Decrypt$  function includes a child function  $RSU.DecryptNode(CT, SK', x)$ , which is defined as a recursive algorithm.

When the  $RSU.Decrypt$  function works, it faces two situations:

1)  $x$  is a leaf node of  $\mathcal{T}$ . We let  $a_i = att(x)$ . If  $a_i \notin \mathcal{S}$ , we get  $RSU.DecryptNode(CT, SK', x) = \perp$ . Otherwise, we get:

$$RSU.DecryptNode(CT, SK', x) = e(D_i, C_x) \\ = e(g^{\beta\gamma v_i^{-1}}, g^{v_i q_x(0)}) \\ = e(g, g)^{\beta\gamma q_x(0)}$$

2)  $x$  is a non-leaf node.  $RSU.DecryptNode(CT, SK', x)$  is worked as same that: for all nodes  $z$  that are children of  $x$ , we call another process  $RSU.DecryptNode(CT, SK', z)$  and let  $F_z$  as output. If  $\mathcal{S}_x$  is an arbitrary  $k_x$ -sized set of child nodes  $z$ , we get  $F_z \neq \perp$ . If  $z$  does not exist,  $F_z = \perp$ . Then, we can compute and return the result as:

$$F_x = \prod_{z \in \mathcal{S}_x} F_z^{\Delta_{i, S'_x(0)}}, \quad \text{where } \begin{cases} i = index(z) \\ S'_x = \{index(z) : z \in \mathcal{S}_x\} \end{cases} \\ = \prod_{z \in \mathcal{S}_x} \left( e(g, g)^{\beta\gamma q_z(0)} \right)^{\Delta_{i, S'_x(0)}} \\ = \prod_{z \in \mathcal{S}_x} \left( e(g, g)^{\beta\gamma q_{parent(z)}(index(z))} \right)^{\Delta_{i, S'_x(0)}} \\ = \prod_{z \in \mathcal{S}_x} \left( e(g, g)^{\beta\gamma q_x(i)} \right)^{\Delta_{i, S'_x(0)}} \\ = e(g, g)^{\beta\gamma q_x(0)}$$

The decryption algorithm on the root node  $R$  of  $\mathcal{T}$  is defined as:

$$F_R = RSU.DecryptNode(CT, SK', R) \\ = e(g, g)^{\beta\gamma q_R(0)} \\ = e(g, g)^{\beta\gamma s_1}$$

$RSU$  computes  $F_R$  if  $\mathcal{S}$  can satisfies the access tree  $\mathcal{T}$ . After that,  $RSU$  continually computes:

$$A = \frac{e(D_1, C_3)}{e(D_2, C_4)} = \frac{e(g^\gamma h^\varepsilon, g^{\beta s_1} \times g^{\beta s})}{e(g^\varepsilon, h^{\beta s_1} \times h^{\beta s})} = e(g, g)^{\beta\gamma(s_1+s)} \\ B = A/F_R = \frac{e(g, g)^{\beta\gamma(s_1+s)}}{e(g, g)^{\beta\gamma s_1}} = e(g, g)^{\beta\gamma s}$$

When  $RSU$  completes the calculation of  $B$ , it can structure the intermediate ciphertext  $CT' = \{CT_f, C_1 = K \times$



$e(g, g)^{\alpha\beta S}$ ,  $C = g^S, B$ ). Finally, *RSU* sends  $CT'$  to *DU* and completes the decryption algorithm.

$DU.Decrypt(CT', SK) \rightarrow K$ . When *DU* accepts  $CT'$  from *RSU*, it puts  $SK$  into the encryption algorithm. The algorithm computes  $SK$  and  $CT'$  as:

$$\frac{C_1 \times B}{e(D, C)} = \frac{K \times e(g, g)^{\alpha\beta S} \times e(g, g)^{\beta\gamma S}}{e(g^{(\alpha+\gamma)\beta}, g^S)} = K$$

*DU* can easily compute  $K$ . Because most of the calculation work has been completed by *RSU*. In this way, *DU* can quickly access the key  $K$  and decrypt the encrypted data.

## B. SMART CONTRACT DESIGN

Smart contracts are the core modules of the blockchain network in FADB. It manages user registration, transaction records and data validation in VANET. Programmatically, a variety of different functions can be deployed for smart contracts to provide continuous scalability for FADB system. In this section, we elaborate on the workflow and intrinsic logic of each functional module in smart contract. Our work is based on the Ethereum client. In Ethereum, smart contracts are programmed using the Turing-complete Solidity language, which provides general tool functions for getting block information and transaction data.

The functions and variables covered in this paper exist in the global namespace of the smart contract in a predefined way. Here are some of the main variables for designing smart contracts:

*tx.origin*: we can get the originator of transaction by calling *tx.origin*. In Ethereum, smart contracts call each other to form a chain of calls. Finally, the *tx.origin* gets the originator of chain of calls.

*msg.sender*: we can get the sender address of current message by calling *msg.sender*. By deploying or invoking smart contracts, the system can get the corresponding user address.

### 1) USER MANAGEMENT CONTRACT

User management contract is a specific implementation of  $SC_{ic}$ . It mainly provides the following functional interfaces to implement user management operations:

*addUser* ( $ID_{driver}, S$ ): This function is executed when the *RSU* calls  $SC_{ic}$  for user registration. It first verifies that the user identity of the contract is correct, and the *RSU* is a working blockchain node in the network. After the verification is passed, it searches on the  $IC$  based on the serial number of the registered device, verifying that the device has already been registered. After that, it reads the  $MSK$  and runs the  $Register(MSK, S)$  to generate the user's private key  $SK$ . After  $SK$  is successfully generated, it constructs a transaction and scatters into the transaction pool  $Tx.pool$ . Finally, the  $SK$  and  $UID$  are passed back to *RSU* over the secure channel.

*getUser* ( $UserID$ ): This function is used to retrieve user's registration information from  $IC$  via  $UID$ . It is called by  $SC_{ic}$  for protecting user information security. There are two ways to retrieve the search method in the function. One is to search by the device number  $ID_{driver}$ . This method is suitable for

---

### Algorithm 1 addUser

---

**Input:**  $ID_{driver}, S \Rightarrow (a_1, a_2, \dots, a_m)$   
**Output:**  $SK, UID$

- 1 **if**  $isRSU(msg.sender) = False$  **then**
- 2     **throw:** *illegal address*
- 3 **if**  $indexUser(ID_{driver}) = True$  **then**
- 4     **throw:** *user already register*
- 5  $Gblock = genesis() \setminus \setminus get$  the genesis block
- 6  $MSK = Gblock.MSK$
- 7  $SK = Register(MSK, (a_1, a_2, \dots, a_m))$
- 8  $Tx = < ID_{driver}, MSK, msg.sender, S, TimeStamp >$
- 9  $SignTx(msg.sender, type.Tx, IC)$
- 10 **while**  $submitTransaction(Tx)$  **do**
- 11     Wait until transaction successfully submitted
- 12  $UID = getUID(Tx_{id})$
- 13 **return:**  $UID, SK$

---



---

### Algorithm 2 getUser

---

**Input:**  $ID_{driver}$  or  $UID$   
**Output:**  $ID_{driver}, SK, S, RSU$

- 1 **if**  $input$  is  $ID_{driver}$  **then**
- 2     **foreach**  $block \in DataChain$  **do**
- 3          $Array[Tx] = BlockRead(block)$
- 4         **foreach**  $Tx$  **in**  $Array[Tx]$  **do**
- 5             **if**  $ID_{driver} == Tx.driverID$  **then**
- 6                  $Tx \Rightarrow \{ID_{driver}, SK, S, RSU\}$
- 7                 **return:**  $\{ID_{driver}, SK, S, RSU\}$
- 8     **return:** *null*
- 9 **else if**  $input$  is  $UID$  **then**
- 10      $block = getBlock(UserChain, UID.block_{num})$
- 11      $Array[Tx] = BlockRead(block)$
- 12      $Tx = getTransaction(block, UID.Tx_{num})$
- 13      $Tx \Rightarrow \{ID_{driver}, SK, S, RSU\}$
- 14     **return:**  $\{ID_{driver}, SK, S, RSU\}$
- 15 **else**
- 16     **throw:** *wrong input*

---

user registration. At this time, the user has not obtained  $UID$ . Thus, the user can only be confirmed by traversing the  $IC$ . The other way is to use  $UID$  for the search. It is easier than the way of searching by  $ID_{driver}$ . User information can be read directly from the  $IC$  based on the block number and transaction number recorded in  $UID$ .

*updateUser* ( $UID, S_{new}$ ): This function is used to update user information. When the user attribute is changed, the range of data that the user can access will also change accordingly. The input of function is the user's  $UID$  and the new attribute set  $S_{new}$ . First, it retrieves user information on the  $IC$  based on the  $UID$ . If the return is empty, the user does not exist. Otherwise we use  $S_{new}$  to re-run the registration

**Algorithm 3** updateUser

---

**Input:**  $UID, S_{new}$   
**Output:**  $UID_{new}, SK_{new}$

- 1 **if**  $getUser(UID) == null$  **then**
- 2     **throw:** *not exist this user*
- 3 **else**
- 4      $User == getUser(UID)$
- 5      $Gblock = genesis()$
- 6      $MSK = Gblock.MSK$
- 7      $SK_{new} = Register(MSK, S_{new})$
- 8      $Tx = \langle User.ID_{driver}, SK_{new}, msg.sender,$
- 9          $S_{new}, TimeStamp \rangle$
- 10      $Tx.type = update$
- 11      $msg.sender$  sign  $Tx$
- 12     **while**  $submitTransaction(Tx)$  **do**
- 13         Wait until transaction successfully submitted
- 14      $UID_{new} = getUID(Tx_{id})$
- 15     **return:**  $UID_{new}, SK_{new}$

---

algorithm and generate a replacement key  $SK_{new}$  for the user. The next steps are similar to the user add operation.  $SC_{ic}$  constructs a update transaction and writes this transaction to  $IC$ . This can avoid multiple matching results when retrieving users. Finally, the function outputs the  $UID_{new}$  and the key  $SK_{new}$ .

## 2) DATA SHARING CONTRACT

Data sharing contract is deployed to handle related matters such as data sharing. Its corresponding operation object is  $DC$ . The functions defined in the data sharing contract are responsible for publishing data metadata to the blockchain and providing data retrieval functions to the  $DU$ . Below we introduce the functional interfaces provided by data sharing contract.

**Algorithm 4** uploadMetadata

---

**Input:**  $CT, kws, hash, F_{address}$   
**Output:**  $True/False$

- 1 **if**  $kws == \phi$  **then**
- 2     **throw:** *key word set cannot be empty*
- 3 **if**  $CT == null$  or  $F_{address} == null$  **then**
- 4     **throw:** *error input*
- 5  $Tx \leftarrow \{CT, kws, hash, F_{address}\}$
- 6  $msg.sender$  sign  $Tx$
- 7  $TxPool.add(Tx)$
- 8 broadcast  $Tx$  in  $TxPool$
- 9 **if**  $Tx$  submits successfully **then**
- 10     make mapping  $kws \rightarrow Tx_{address}$
- 11     **return:**  $True$
- 12 **return:**  $False$

---

$uploadMetadata(CT, kws, hash, F_{address})$ : This function provides a convenient data write interface that allows users to

**Algorithm 5** queryData

---

**Input:**  $UID, keyword$   
**Output:**  $SK', CT, hash, F_{address}$

- 1 call *User Management Contract*
- 2  $user \leftarrow getUser(UID)$
- 3 **if**  $user == null$  **then**
- 4     **throw:** *Unauthorized Access*
- 5 **else**
- 6      $SK' = keyCat(user.SK)$
- 7  $addressList \leftarrow mapping(keyword)$
- 8 **if**  $addressList == null$  **then**
- 9     **return:**  $null$
- 10 **else**
- 11     **foreach**  $Tx_{address}$  in  $addressList$  **do**
- 12          $Tx = getTx(Tx_{address}, DataChain)$
- 13          $Tx \Rightarrow \{CT, kws, hash, F_{address}\}$
- 14          $result.add(CT, hash, F_{address})$
- 15     **return:**  $SK', result$

---

quickly write metadata information to the  $DC$ . First, it checks the input. If the input is wrong, it throws the corresponding exception. Otherwise, it constructs a corresponding transaction and signs the transaction. Then, we store the transaction in  $Tx.pool$  and wait this transaction to be written into the  $DC$ . The transactions of  $Tx.pool$  are broadcasted to the whole blockchain network, allowing all  $RSUs$  to confirm the legality of transactions. When the blockchain network has accumulated a large of transactions that can be accommodated by a block, the transactions are packaged in bulk and written into a new block. After the new block is successfully backed up by all nodes, it indicates that the metadata is successfully written into the  $DC$ .

$uploadMetadata(UID, keyword)$ : This function retrieves data quickly on the  $DC$  via  $kws$ . After checking the correctness of  $kws$ , it checks the validity of user by calling the  $getUser$  function. And then it truncates the  $SK$  for subsequent CP-ABE decryption. After that, it realizes the address mapping according to  $kws$  and gets the  $addressList$ . For zero address in  $addressList$ , the function  $queryData$  returns a null value. For multiple addresses, the function accesses each address and obtains the data metadata stored in the corresponding transaction. Finally, all retrieved metadatas are encapsulated in an array and returned with the key  $SK'$  to the requested  $RSU$ .

**VI. EVALUATION**

In this section, we simulate FADB under various conditions. Then we analyze and contrast the results obtained by simulation. Because the FADB introduces new components, blockchain and HECP-ABE. They does not exist in current VANET. Thus, it brings unknown performance impact and cost for VANET environment. We need test the blockchain

network and the HECP-ABE separately to evaluate the performance of FADB in actual deployment.

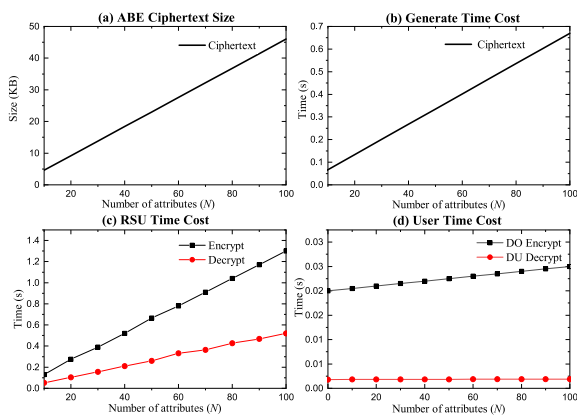
**A. EXPERIMENT ENVIRONMENT**

The simulation is done on an Ubuntu 16.04.4 LTS desktop equipped with an Intel Core i7-8086K @4.0GHz processor and 16G of RAM. We use Docker to virtualize a physical machine into multiple virtual machines, and configure different hardware resources to simulate *RSU* and *OBUE* devices in FADB. On the basis of the Ethereum ganache-cil, a certain repair is carried out to meet the needs of our simulation. The encryption scheme uses the CP-ABE encryption toolkit and uses the PBC library for algebraic operations which provides four command-line tools for performing various operations. An 80-bit security level is achieved by using a 160-bit elliptic curve group.

**B. PERFORMANCE**

1) ENCRYPTION CONSUMPTION

In HECP-ABE, the complexity of the calculation is related to the size of the user attribute set. The increase of attributes leads to a complex ciphertext strategy. This affects the time of encryption and decryption and the final ciphertext length. To test how our solution is affected by the size of attribute set, we constructed 100 different size of attribute set. The number of attributes in each set is increased from 1 to 100. Finally we generate a key for each set.



**FIGURE 7. Encryption consumption.**

Figure 7 is based on the number of attributes encapsulated in the key in our scheme. It can be clearly seen that the length of the ciphertext is more positively related to the number of attributes.

For Figure 7(a), we can get that the resulting ciphertext size is about 47 KB under the ciphertext policy with the attribute set, which contains 100 attributes.

For Figure 7(b), the initialization phase of the HECP-ABE algorithm is not so slow, and it takes about 0.66 seconds to generate the system master key in the case of 100 attributes.

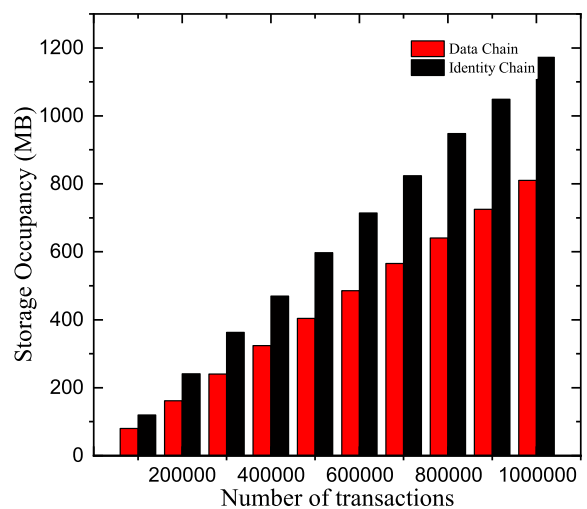
For Figure 7(c), as the number of attributes increases, the *RSU* time cost increases linearly for executing

encryption/decryption algorithm. But the time cost growth rate of the decryption algorithm is smaller than the encryption algorithm. Under the premise of 100 attributes, it takes about 1.33 seconds and 0.45 seconds for encryption and decryption respectively.

For Figure 7(d), compared with *RSU* in Figure 7(c), user (*DO/DU*) time cost is faster in encryption and decryption. *DO* needs about 0.025 seconds to encrypt the ciphertext policy containing 100 attributes. While *DU* needs 1.9 milliseconds to decrypt the ciphertext. Because the decryption process of *DU* is basically unaffected by the number of attributes contained in the ciphertext policy. As expected, our solution splits the encryption and decryption process by transferring most of the encryption and decryption operations to the high-performance *RSU*. The time on VANET devices in the process of encryption and decryption is greatly reduced. Compared with the uploading and downloading of data, the entire encryption and decryption process does not take up too much time cost. Thus, we can get that our CP-ABE scheme is applicable to FADB.

2) STORAGE COST

Due to the introduction of blockchain in VANET, it is inevitable that blockchain can result in more storage usage. In order to evaluate the additional storage usage caused by maintaining two chains (*IC* and *DC*) in FADB, we simulate the block growth in real-world situations by simulating transactions. We build a blockchain network with *RSU* nodes. The reason for constructing only *RSU* nodes is to reduce the number of simulation steps and to shorten the simulation time. In the simulation, the *RSU* node directly constructs the transactions of user registration and data upload. And we simulate real transactions by random functions. Figure 8 shows the storage occupancy of the *IC* and *DC* under 1-1,000,000 transactions.



**FIGURE 8. Storage comparison.**

It can be seen from the Figure 8 that the overall storage occupancy grows linearly and is positively correlated with the

number of transactions. This is because that the size of each transaction in FADB is fixed, which is different from traditional Ethereum. Each transaction in traditional Ethereum has a data field. This field allows for the embedding of external information, which makes the size of each transaction not fixed. However, in our FADB, the transaction is dedicated to user registration and data management. So the size of each transaction is fixed. The growth rate of storage occupancy in *DC* is smaller than *IC*. Because the size of the user attribute set is an unknown quantity, and the larger attribute set leads to an increase in the space occupied by the transaction. It causes the *IC* to take up more storage. Under 1,000,000 transactions, the *IC* occupies 1172.3MB of storage space. While the *DC* occupies 810.3MB of storage space. *IC* and *DC* together occupy 1982.6MB of storage space. Suppose that the entire blockchain network generates 100 transactions per second in real situation. Then the entire network produces 17129MB of block files a day, which is only a small value relative to the shared VANET data volume. For multiple machines, FADB's storage consumption is much lower than expected. Because the number of *DO/DUs* is much larger than the RSUs on multiple machines. *DO/DUs* do not participate in the storage of blocks. Blocks are only stored in RSUs. This significantly reduces storage consumption per machine. And the storage consumption does not increase significantly as the number of machines increases.

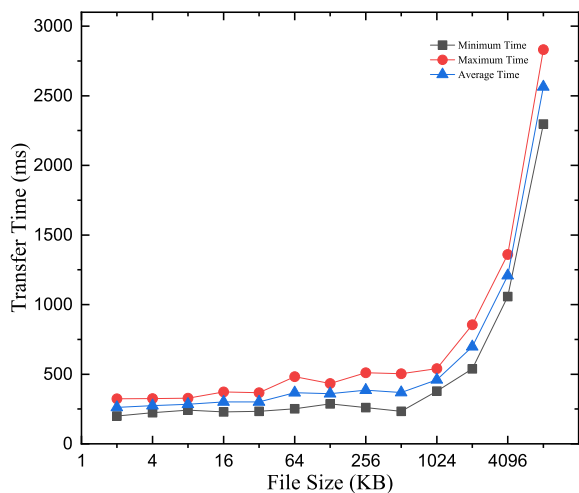


FIGURE 9. Performance of small file transfer.

### 3) TRANSMISSION RATE

We test the transmission rate of sharing data files under different file sizes in FADB. We define the files below 10MB as small files. Figure 9 shows the test results for small files. It can be seen that the average transmission rate increases exponentially with the increase in file size. This is because the main factors affecting the overall transmission speed are not the physical bandwidth and the network delay. The main factors of time cost are encryption and decryption of files, the storage of metadata in blockchains, and the retrieval of

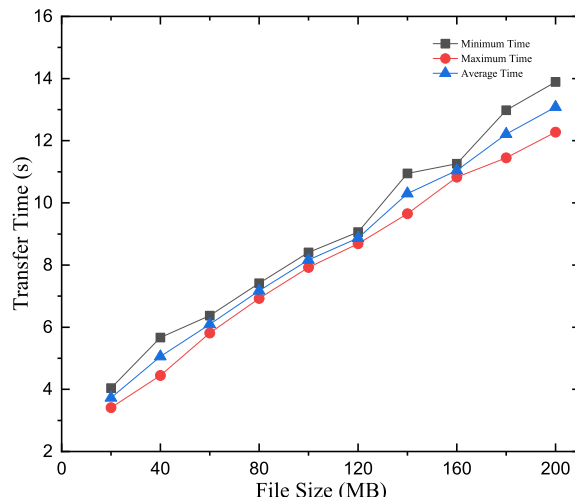


FIGURE 10. Performance of large file transfer.

files. The consumption of this part is fixed and much longer than the consumption of file transfer. So the growth of the front part of the curve is very flat. When the file exceeds a certain size, network delay and physical bandwidth are the main factors, as shown in Figure 10. In the case of large file sharing, the time consuming of the whole process is basically linear with the size of the file. The larger the transmission bandwidth, the faster the transmission speed. But the occasional fluctuations are related to network delay and file retrieval.

## VII. CONCLUSION

In this article, we introduce the design and implementation of a new data sharing architecture called FADB. By combining blockchain technology, IPFS distributed storage, and CP-ABE encryption, the FADB provides a data sharing platform that integrates data security, privacy protection, and authorized access. FADB seamlessly accesses VANET to provide users with reliable data storage and sharing services.

The FADB contains a blockchain network consisting of RSUs. Two blockchains (*IC* and *DC*) are maintained in FADB. They are responsible for managing the user identity information and shared data. All data on the chains is read/written and maintained through smart contracts. All RSUs have all block backups, which makes user information and metadata recoverable and can effectively resist external attacks. In FADB, IPFS can effectively avoid single points of failure compared to traditional cloud storage solutions. It uses replication proof, erasure coding and incentives to provide better reliability and availability.

We have also implemented a new type of efficient encryption scheme HECP-ABE in FADB. It combines the traditional CP-ABE encryption scheme with the blockchain, which makes our solution can provide a distributed, fine-grained data sharing service. By using HECP-ABE encryption scheme, the *DO* can restrict access to the data for a specific user by establishing an access policy. Thus, it enables the

fine-grained access control. In HECP-ABE, we have split the encryption and decryption steps, by transferring most of the calculation operations to the *RSU*. It effectively reduces the computational pressure of the lightweight device in VANET.

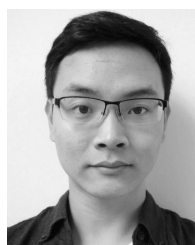
In future work, we will further optimize the retrieval function of VANET data on blockchain, implement the user attribute revocation function in HECP-ABE scheme. Furthermore, we will consider more data security protection features (e.g. level of anonymity, stateless access, etc.) and strengthen experiments to prove the effectiveness of security protection.

## REFERENCES

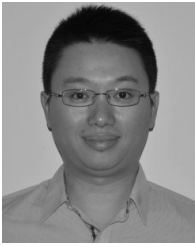
- [1] H. Hartenstein and K. Laberteaux, *VANET Vehicular Applications and Inter-Networking Technologies*, vol. 1. Hoboken, NJ, USA: Wiley, 2010.
- [2] C. Chatrpathi, M. N. Rajkumar, and V. Venkatesakumar, "VANET based integrated framework for smart accident management system," in *Proc. Int. Conf. Soft-Comput. Netw. Secur. (ICSNS)*, Feb. 2015, pp. 1–7.
- [3] S. Bitam, A. Mellouk, and S. Zeadally, "VANET-cloud: A generic cloud computing model for vehicular ad hoc networks," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 96–102, Feb. 2015.
- [4] G. Greenwald and E. MacAskill, "NSA Prism program taps into user data of Apple, Google and others," *Guardian*, vol. 7, no. 6, pp. 1–43, 2013.
- [5] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, "A survey on privacy protection in blockchain system," *J. Netw. Comput. Appl.*, vol. 126, pp. 45–58, Jan. 2019.
- [6] K. Ashokkumar, B. Sam, and R. Arshadprabhu, "Cloud based intelligent transport system," *Procedia Comput. Sci.*, vol. 50, pp. 58–63, Jan. 2015.
- [7] R. Koduri, S. Nandyala, and M. Manalikandy, "Secure vehicular communication using blockchain technology," SAE Tech. Paper 2020-01-0722, 2020.
- [8] S. S. Vattaparambil, R. Koduri, S. Nandyala, and M. Manalikandy, "Scalable decentralized solution for secure vehicle-to-vehicle communication," SAE Tech. Paper 2020-01-0724, 2020.
- [9] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. Berlin, Germany: Springer, 2013.
- [10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
- [11] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2005, pp. 457–473.
- [12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 89–98.
- [13] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.
- [14] J. Benet, "IPFS—content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*. [Online]. Available: <http://arxiv.org/abs/1407.3561>
- [15] S. Nakamoto et al. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Bitcoin. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [16] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
- [17] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, p. 8, Sep. 2014.
- [18] B. Xu, D. Luthra, Z. Cole, and N. Blakely. (2018). EOS: An Architectural, Performance, and Economic Analysis. Bitmex. [Online]. Available: <https://www.whiteblock.io/library/eos-test-report.pdf>
- [19] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016.
- [20] J. Kishigami, S. Fujimura, H. Watanabe, A. Nakadaira, and A. Akutsu, "The blockchain-based digital content distribution system," in *Proc. IEEE 5th Int. Conf. Big Data Cloud Comput.*, Aug. 2015, pp. 187–190.
- [21] S. Kiyomoto, M. S. Rahman, and A. Basu, "On blockchain-based anonymized dataset distribution platform," in *Proc. IEEE 15th Int. Conf. Softw. Eng. Res., Manage. Appl. (SERA)*, Jun. 2017, pp. 85–92.
- [22] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.
- [23] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 468–475.
- [24] N. Kaaniche and M. Laurent, "A blockchain-based data usage auditing architecture with enhanced privacy and availability," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2017, pp. 1–5.
- [25] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. 17th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2017, pp. 468–477.
- [26] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 735–737.
- [27] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.
- [28] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.
- [29] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Proc. 8th Int. Conf. Netw. Service Manage. (CNSM), Workshop Syst. Virtualization Manage. (SVM)*, 2012, pp. 37–45.
- [30] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2014.
- [31] S. K. Pasupuleti, P. J. A. Alphonse, and P. K. Premkamal, "Efficient revocable CP-ABE for big data access control in cloud computing," *Int. J. Secur. Netw.*, vol. 14, no. 3, p. 119, 2019.
- [32] A.-P. Xiong, Q.-X. Gan, X.-X. He, and Q. Zhao, "A searchable encryption of CP-ABE scheme in cloud storage," in *Proc. 10th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. (ICCWAMTIP)*, Dec. 2013, pp. 345–349.
- [33] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," 2019, *arXiv:1906.11078*. [Online]. Available: <http://arxiv.org/abs/1906.11078>
- [34] L. W. Cong and Z. He, "Blockchain disruption and smart contracts," *Rev. Financial Stud.*, vol. 32, no. 5, pp. 1754–1797, May 2019.



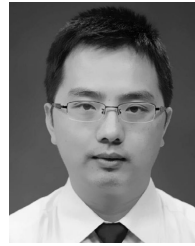
**HUI LI** received the B.S. and M.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan, China, in 2012 and 2016, respectively, where she is currently pursuing the Ph.D. degree in information and communication engineering.



**LISHUANG PEI** received the bachelor's degree from the Chengdu University of Technology (CDUT), in 2018. He is currently pursuing the master's degree in electronics and communications engineering from UESTC, Chengdu, Sichuan, China.



**DAN LIAO** is currently a Professor with the University of Electronic Science and Technology of China (UESTC). His research interests include next generation networks, and wired and wireless computer communication networks and protocols



**MING ZHANG** is currently a Senior Engineer with the Chengdu Research Institute, University of Electronic Science and Technology of China (UESTC). His research interests include the Internet of Things, blockchain, embedded intelligent control, and high-performance motion control. He has coauthored ten technical publications including articles in refereed journals, conferences, and book chapters.



**SONG CHEN** received the bachelor's and master's degrees in computer science from the University of Electronic Science and Technology of China. He is currently a Senior Engineer and Expert with the No. 30 Research Institute of China Electronic Technology Corporation (CETC). He presided over many advanced projects in communication and network field, including pre-study and model development ones. His research interests include network switching and routing, software defined networks, and network security.



**DU XU** is currently a Professor with the University of Electronic Science and Technology of China (UESTC), Chengdu, China. He presided over many advanced research projects, including NSFC, National 863 Plans, and National Key Research and Development Program of China.

• • •