IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 7, 2019, accepted March 5, 2019, date of publication March 11, 2019, date of current version March 29, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2904042*

# Privacy-Oriented Blockchain-Based Distributed Key Management Architecture for Hierarchical Access Control in the IoT Scenario

**MINGXIN MA[1], (Student Member, IEEE), GUOZHEN SHI[2], AND FENGHUA LI[3], (Member, IEEE)**

[1]School of Cyber Engineering, Xidian University, Xi'an 710071, China
[2]School of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China
[3]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Corresponding author: Mingxin Ma (mamx1992@qq.com)

This work was supported in part by the National Key Research Program of China under Grant 2017YFB0801803, Grant 2016YFB0800304, and Grant 2017YFB0802705.

**ABSTRACT** The rapid development of the Internet of Things (IoT) and the explosive growth of valuable data produced by user equipment have led to strong demand for access control, especially hierarchical access control, which is performed from a group communication perspective. However, the key management strategies for such a future Internet are based mostly on a trusted third party that requires full trust of the key generation center (KGC) or central authority (CA). Recent studies indicate that centralized cloud centers will be unlikely to deliver satisfactory services to customers because we place too much trust in third parties; therefore, these centers do not apply to user privacy-oriented scenarios. This paper addresses these issues by proposing a novel blockchain-based distributed key management architecture (BDKMA) with fog computing to reduce latency and multiblockchains operated in the cloud to achieve cross-domain access. The proposed scheme utilizes blockchain technology to satisfy the decentralization, fine-grained auditability, high scalability, and extensibility requirements, as well as the privacy-preserving principles for hierarchical access control in IoT. We designed system operations methods and introduced different authorization assignment modes and group access patterns to reinforce the extensibility. We evaluated the performance of our proposed architecture and compared it with existing models using various performance measures. The simulation results show that the multiblockchain structure substantially improves system performance, and the scalability is excellent as the network size increases. Furthermore, dynamic transaction collection time adjustment enables the performance and system capacity to be optimized for various environments.

**INDEX TERMS** Blockchain, fog computing, hierarchical key management, Internet of Things.

## I. INTRODUCTION

The Internet of Things (IoT) is an extended network based on heterogeneous networks that provides strong support for the interconnection of thing-to-thing (T2T), human-to-thing (H2T) and human-to-human (H2H) interactions. With billions of "things" included, IoT is considered to be the future Internet, providing a worldwide network of interconnected objects that can "feel" (gather physical data), "think" (process data in a fixed or intelligent manner), and "talk" (communicate with other entities using a wired or wireless channel) [1]. According to a survey [2] organized by the Eclipse IoT Working Group and IEEE IoT, security is the greatest concern in the development of IoT solutions; communication security and data encryption are the most popular techniques to achieve IoT security.

The enormous amount of data gathered by IoT devices is of great value for Big Data mining, statistics and analysis. However, this promise relies on the accessibility of a massive amount of data that resides in the cloud and at the edge (e.g., sensors) and is owned by IoT users [3]. Current models for data sharing and user privacy are commonly based on the trustworthiness of the trusted third party, which stores data securely and achieves access control by maintaining an access control list (ACL). The key management schemes based on the preshared key framework and key

The associate editor coordinating the review of this manuscript and approving it for publication was Kuan Zhang.

pool framework are operated at the key generation center (KGC) and are not scalable for large numbers of entities and dynamic changes in relationships. The hierarchical key assignment scheme (HKAS) can be used to solve the hierarchical multigroup communication problem by allowing authorized users to have different access privileges. However, complicated relationships, e.g., when one of the parties is not only the subject to access but also the object to be accessed by the same entity, are beyond the scope of HKAS. In addition, a central authority (CA) with full trust is also needed in HKAS. Therefore, these schemes are not suitable for privacy-oriented scenarios. For instance, electronic health record (EHR) systems and smart homes represent groups of users who are sensitive to the right to control their own data and are reluctant to hand over the initiative of key management to a third party, which may compromise user privacy.

The idea of the blockchain used in cryptocurrencies is merged into our work to eliminate the drawback of introducing a third party. The blockchain technique is the backbone of Bitcoin proposed by S. Nakamoto in ''Bitcoin: A peer-to-peer electronic cash system'' [4]. With the development and popularity of cryptocurrency, researchers have shifted their attention to supporting technologies. The main idea of the blockchain is distributing decision-making operations from a centralized organization to all participants, eliminating the connotative challenges brought by the trusted third party. Simultaneously, a successful implementation makes full use of the computing resources and storage resources of all participants, which improves efficiency and resilience. In addition, the traceability and auditability of the blockchain provide an effective approach to audit key management and data sharing throughout the full lifecycle.

In this paper, we propose a privacy-oriented blockchain-based distributed key management scheme to achieve hierarchical access control. By introducing the blockchain to IoT, we eliminate potential challenges brought by the trusted third party. When using the blockchain to achieve hierarchical access control, various questions, such as those regarding where verified relationships are stored, who is responsible for maintaining dynamic relationships, and why the relationships are trusted, must be addressed. A novel blockchain concept is introduced into the proposed scheme to manage and maintain relationships and to determine and authorize access queries without a trusted third party. The user equipment (UE) declares its relationships while participating in the network and inherits more relationships dynamically and automatically in a specified manner. The blockchain is operated by security access managers (SAMs) who play the role of the CA. The logical topology is stored in the SAMs, and key management operations are stored in blockchains that act as public ledgers. Furthermore, to satisfy the low-latency and high-scalability requirements of the IoT scenario, we introduce cloud managers to operate multiblockchains composed of different blockchains operated in each deployment domain. We analyze the time consumption, transaction collection period and target difficulty for block mining to dynamically adjust to various transaction quantities.

The rest of this paper is structured as follows. Section II introduces key management techniques and the blockchain. Section III presents an overview of the proposed scheme and discusses it in detail. We then describe our system model, including system operations, transaction and block formats, and time composition. The performance of the proposed scheme is evaluated in Section IV. Section V concludes this paper and presents directions for future research.

## II. RELATED WORK

In this section, we present a brief review of key management schemes (KMS) and the blockchain technique and its applications.

### A. OVERVIEW OF KEY MANAGEMENT SCHEMES

In this section, we focus on traditional strategies to solve hierarchical access control. KMS can be categorized into the following main frameworks [1]: preshared key frameworks [5], key pool frameworks [6], mathematical frameworks [7], [8], public key frameworks (public key cryptography (PKC)) [9], and HKAS [10]–[12]). In a typical prekey algorithm that assigns the same key to each participant, resilience is a major problem because a single-point compromise will lead to destruction of the entire network. Therefore, such a framework cannot satisfy the goal of hierarchical access control. One possible way to achieve this goal is to assign different keys to each group of participants; however, the number of keys increases rapidly with the complexity of the access relationships. In the key pool paradigm, the network designer creates a key pool and assigns every node a unique key chain that helps each pair of nodes find a common shared key path to negotiate a pairwise key. The key chain assigned to each node can be designed to achieve hierarchical access control, but the number of available key chains decreases with each key update and revoke operation, resulting in periodic network-wide scope initialization. Blom *et al.* [7] scheme requires storing only $\lambda + 1$ keys for each node, where $\lambda \ll N$ and $N$ is the number of nodes in the network. The scheme achieves optimal resilience at the expense of a relatively large memory requirement and [8] improves the method for application in resource-constrained environments. This strategy guarantees connectivity and robustness but neither achieves hierarchical access control nor is suitable for scenarios in which keys are frequently updated and nodes are revoked. A common disadvantage of these strategies is that they can neither distinguish the source of the access query nor achieve accurate access records. Moreover, they lack the capability to update keys in asynchronous patterns for multiple devices.

In the hierarchical key management architecture, users are organized in a hierarchy and divided into separate groups according to access privileges. Public information must be published either commonly for all nodes [10], [11] or individually for each node [12] to realize hierarchical access control. However, complicated relationships, e.g., where one of the parties is not only the subject to access but also the object to be accessed by the same entity, are beyond the reach of HKAS.

Public key infrastructure (PKI) has matured and is now widely implemented, especially methods based on elliptic curve cryptography (ECC). Despite their high computational complexity, PKC strategies have some good properties. They offer good resilience, attackers can impersonate only certain compromised devices, preventing further leakage, and they promise to achieve accurate access records. Many services, such as registration authority (RA), certificate authority (CA), certificate revocation list (CRL), and online certificate status protocol (OCSP), are attached to the PKI to enhance its performance. However, the certificate verification cannot be accomplished without an intermediate CA and a root CA; therefore, the requirements of privacy-oriented scenarios mentioned before cannot be satisfied when the trustworthiness of CA is in doubt, especially when the public/private key pair is generated by the CA.

Notably, these strategies are all centralized KMS: even if deployed in a distributed manner, the key generation procedure and/or verification procedure is based on the trustworthiness of the third party. Therefore, none of these strategies satisfies the requirements of privacy-oriented scenarios where users are sensitive to the right to control their own data and reluctant to hand over the initiative of key management to a third party.

### B. BLOCKCHAIN AND ITS APPLICATIONS

The core contribution of the blockchain is the maintenance of a distributed, authenticated, and synchronized ledger of transactions for all participants [13]. A successful implementation makes full use of the available computing resources and storage resources of all participants, which improves the efficiency and resilience and results at a low cost compared to cloud computing [14]. Benefiting from the materials and manufacturing process, public-key strategies are viable for IoT devices [15], [16]; moreover, resource constraints are much looser for intelligent devices.

A blockchain has three main characteristics: 1) decentralization, 2) traceability, and 3) persistency. Authenticated transactions in blocks are recorded in a decentralized manner, where miners compete for the rights to write the new block, and the motivation to maintain the consensus of the blockchain is to gain rational income. Decentralization management avoids a single point of failure; by contrast, centralized managers suffer from excessive communication and computation problems. Benefiting from the chain form of blocks, a KMS using blockchain provides timely revoking and updating functionalities in an asynchronous pattern. Traceability is another advantage of the blockchain. Every transaction must refer to previous unspent transactions; therefore, the value flow and information flow are easy to trace, and the validity of the transactions is easily verified. Therefore, the blockchain provides an efficient way to record a key management log for ease of audit. The security and privacy of the network are based on consensus algorithms. Many consensus algorithms, such as proof-of-work [17], proof-of-stake [18], delegated-proof-of-stake [19], and byzantine fault tolerance [20], [21], have been proposed. The greater the decentralized rights to create and record transactions, the more immutable and steady the system; in contrast, propagation and verification efficiencies decrease with increasing decentralization. However, the profit-driven nature of the market allows users to form mining pools to obtain greater revenue. According to probability and statistics, it would take 5–10 years for a general user to find a block, and when participating in the mining pools, user could suffer a fork after withholding (FAW) attack [22], which is more serious and practical than selfish mining [23]. References [24] and [25] precisely analyzed the security threats to blockchain systems.

With the development and popularity of cryptocurrency, blockchain has received substantial attention, and schemes adopting the blockchain concept have been proposed in many research fields [26]. One study [27] introduced a decentralized, peer-to-peer platform called BPIIoT to provide secure, auditable, and autonomous manufacturing services. Lei *et al.* [28] proposed the use of blockchain to build a heterogeneous intelligent transportation system (HITS), encapsulate cryptographic materials into the block and propagate them to a security manager (SM) cloud. The authors compared the network performance, in terms of key transfer time, with that of schemes with a central manager. However, they did not consider the propagation procedure of SM clouds. Dorri *et al.* [29] proposed a hierarchical architecture consisting of smart homes, an overlay network and cloud storage to provide privacy and security. The design used different types of blockchains depending on the location of the transaction in the network hierarchy and distributed trust methods to ensure a decentralized topology. However, they present an abstract conception without experimental or simulation results and further research. Reference [30] focused on using blockchain technology to solve the decentralized management of private data and the digital property resolution of Big Data. Blockchain can influence the way in which Big Data is used to find a solution for storing and managing data in a distributed manner on a peer-to-peer (P2P) network.

Sharma *et al.* [31] proposed a flexible, efficient, scalable, and securely distributed cloud architecture with blockchain and fog computing. By bringing computing resources to the edge of the IoT network, the system has minimal end-to-end delay between IoT devices and computing resources. Reference [32] moved IoT components from the cloud to edge hosts to reduce overall network traffic and minimize latency. The experimental results show that network latency is the dominant factor in system efficiency; therefore, deploying the blockchain-based key management system on the fog, which is closer to terminal devices, is beneficial to the IoT scenario.

To the best of our knowledge, the only previous studies related to our solution are Ouaddah *et al.* [33] and Novo [34].

Ouaddah *et al*. [33] described a cryptocurrency blockchain-based access control framework called FairAccess and created smart contracts for the access control policy of every resource-requester pair to achieve access control. Oscar introduced another way to achieve access control by creating a single smart contract to define the policy rules of the management system and release the computational overhead of IoT devices. In contrast, we focus on privacy-oriented IoT scenarios and achieve hierarchical access control in an individual manner. The access control policies are converted to registration and update messages to form a logical topology and are assigned not only by UE itself but also inherited from different authorization assignment modes. Furthermore, by using a group access pattern, a subject who wishes to access a set of UEs is authorized through a single access query.

## III. BLOCKCHAIN-BASED DISTRIBUTED KEY MANAGEMENT ARCHITECTURE FOR THE IoT

### A. SYSTEM MODEL

#### 1) ARCHITECTURE DESIGN OVERVIEW

We focus exclusively on IoT systems that prefer to keep the user's key information, such as individual smart homes, secure and out of the KGC's control. Figure 1 presents an overview of the architecture of the proposed model, which is composed of a cloud layer, a fog layer, and a device layer connected with core networks and edge networks. In the device layer, UE, including wireless sensors, surveillance cameras, smart bracelets, controllers, and individual medical devices, is used to sense, monitor, and control the surrounding environment. Edge networks provide the access function for the devices, such as 5G base stations, Ethernet, and WiFi. In the fog layer, the security access manager, which has some computational capabilities, is used to record and verify transactions that include key management information, such as smart gateways that remain permanently online. The proposed key management blockchain is operated on SAMs to provide a low-latency key management function for UE in the same deployment domain. The core networks provide high-speed service to connect the SAM to the cloud. The cloud layer has multiblockchains composed of each blockchain from the fog layer. The tremendous computational capability of the cloud helps achieve interconnection and traceability among blockchains, that is, cross-domain interaction. Devices that join the blockchain on the fog can act as light nodes, e.g., UE that processes only self-correlative key information transactions, or full nodes, e.g., SAMs that store and verify whole transactions of the blockchain. The blockchain is joined according to the deployment tuple consisting of the application field and location to prevent consumption for calculating and storing irrelevant transactions and to reduce the latency caused by transmission. Meanwhile, the cloud can also store the encrypted data generated by wireless sensors, surveillance cameras, etc., and the data can be obtained directly from the cloud after other devices obtain the encryption key.

#### 2) BLOCKCHAIN-BASED STRUCTURE FOR THE SAME DOMAIN

As presented in Figure 2(a), SAMs act as network managers to maintain the key information blockchain of their domain. Each SAM is connected with several UEs to handle their cryptographic operations encapsulated into transactions, and SAMs are connected to each other to synchronize the public ledger. SAMs play the role of miners to record key information transactions in the ledger and package the transactions of UE within a specified period of collection time into a new block. The key information transactions that must be recorded in the blockchain are divided into the following six categories.

##### *a: INITIALIZATION REGISTRATION*

New UEs must apply for initialization registration when they join a blockchain. UE must specify its deployment tuple and encrypted access key (for auditing) and indicate which nodes are authorized to access itself. Meanwhile, to enhance flexibility, the authorization assignment mode is also specified, which we will explain in Section III. B 4).

##### *b: ACCESS QUERY*

To obtain access permission to a certain node, the subject must send an access query transaction to the SAM. After the initialization stage is completed, the SAMs that operate the blockchain have a comprehensive view of the authorized nodes of each device to help them to judge and verify the authority of the particular access transaction.

##### *c: ACCESS RECORD*

After access permission is granted to a subject, the subject decides when or whether to access the object in the key lifetime. Therefore, the access operation launched by the object must be recorded along with the access signature of the subject in the access record transaction. Generally, this point is the end of the access cycle.

##### *d: KEY UPDATE*

For safety reasons, such as key expiration for past authorized nodes, UE must periodically update the access keys; therefore, UE must send a key update transaction to the SAM to record the new key and its lifetime.

##### *e: LEAVE*

Before a node leaves the network, it usually generates a new transaction to declare an incoming action and indicates its own parents to be its children's new parent nodes. This operation is optional, and each node can decide whether to accept this arrangement according to the authorization assignment mode.

##### *f: ADVERSARY REVOCATION*

If a node is discovered to be compromised or malicious, the SAM signs a new transaction and records it into a new block to declare the revocation. Since all transactions

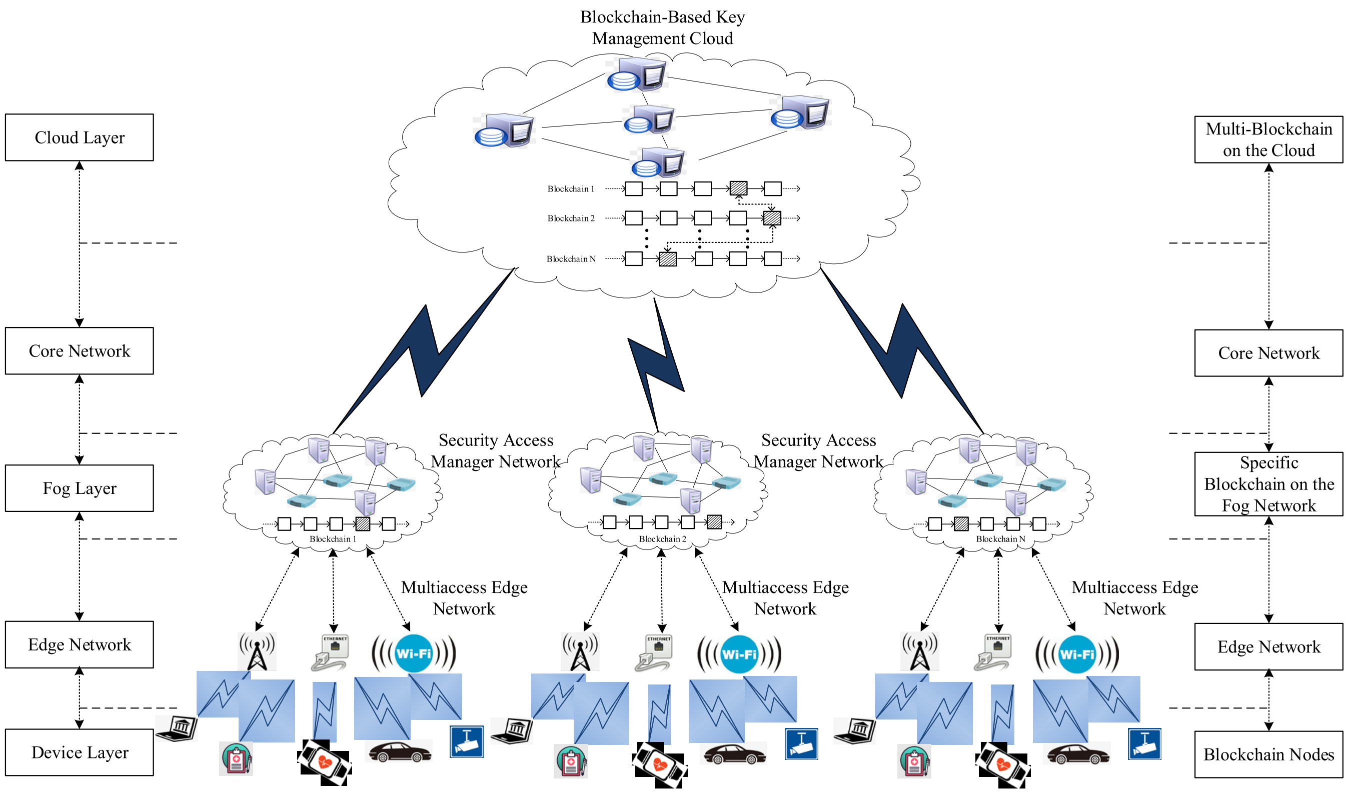


**FIGURE 1.** **Overview of the blockchain-based distributed key management architecture.**

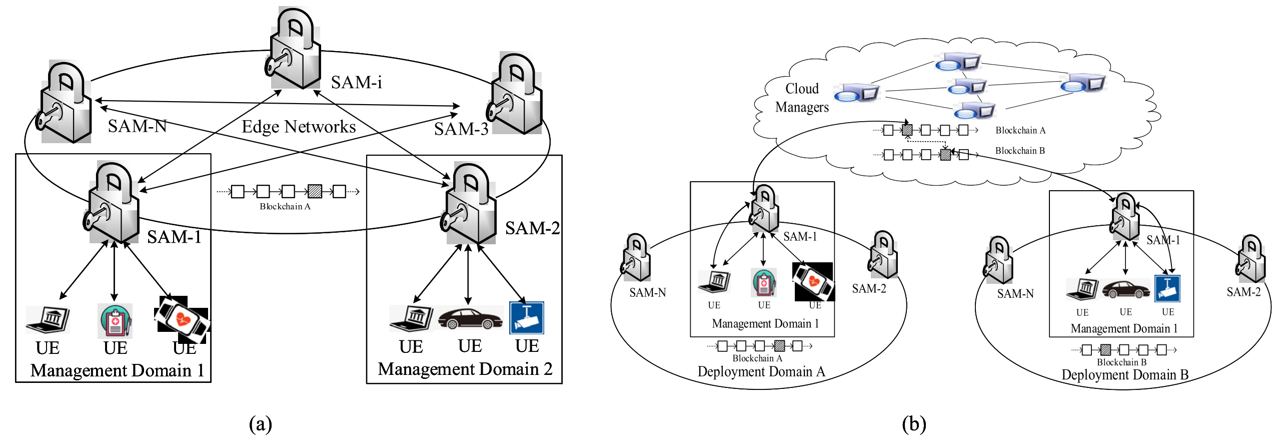


**FIGURE 2.** **Network structures. (a) Blockchain-based structure for the same deployment domain. (b) Blockchain-based structure for the cross-domain.**

with respect to a certain node are linked by the blockchain, the revoked nodes can no longer access (deny) or be accessed (risk warning).

The SAMs record transactions in a new block that is mined periodically by a random SAM who solves the POW at a certain difficulty, which means the target number of zeros $n_{zeros}$ at the start of the hash result of the block header has been found. A trivial solution to recording the transactions for each SAM is to compete for the right to generate the next block or to perform this job by turns, and the behavior is supervised by other SAMs to guarantee legality. However, when the network size increases substantially such that it contains numerous SAMs, an unacceptable average time $n \cdot t_{mine}$ is required to record the transactions for a SAM, where $n$ is the number of SAMs and $t_{mine}$ is the period required to generate a new block. To accelerate this procedure, the transactions of different SAMs are propagated to each other to be recorded simultaneously; therefore, a reasonable collection period must be chosen to balance the collection time, the propagation time, the block mining time and the number

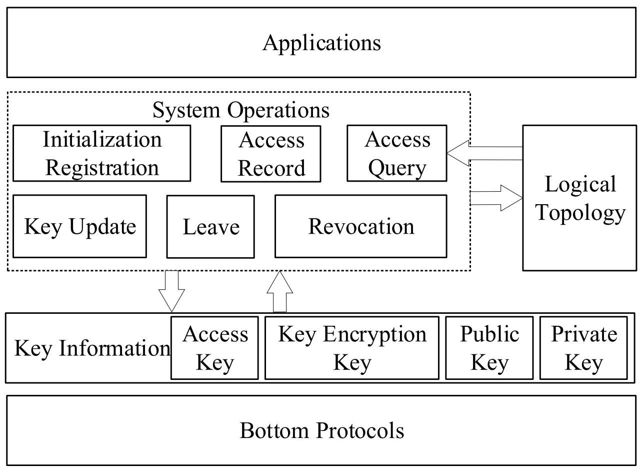


**FIGURE 3.** **Blockchain-based distributed key management framework.**

of transactions in one block. We analyze the time composition in Section III. B 5).

### 3) BLOCKCHAIN-BASED STRUCTURE FOR CROSS-DOMAIN

As mentioned above, IoT applications cover various fields and locations defined by so-called deployment tuples. To limit the number of transactions in one block to a reasonable size within a certain collection period and decrease the number of unnecessary transactions for a group of UEs, we implement several blockchains for different deployments and store them in the cloud in favor of cross-domain interaction.

In the initialization registration of the blockchain-based structure for the same domain, UE specifies its deployment tuple to join the particular domain, i.e., the particular blockchain operated by correlative SAMs. In each blockchain, only correlative transactions are recorded, and a copy of the blockchain is stored in the cloud. Cloud managers, which are similar to SAMs, operate whole blockchains and verify the access authentication when a subject from deployment domain A needs to access an object from deployment domain B. After cross-domain access is verified, the cloud manager signs a transaction to indicate the behavior and sends the transaction back to SAMs in both deployment domains. The above procedures are depicted in Figure 2(b).

## B. KEY MANAGEMENT SCHEME

We illustrate the framework of the proposed blockchain-based distributed key management scheme for IoT in this section. Additionally, we describe the system operations and the blockchain format used to implement our scheme. Moreover, to achieve a lightweight, scalable and adaptive key management scheme, we further analyze the authorization assignment mode and time composition.

### 1) SYSTEM OPERATIONS

As shown in Figure 3, the operations are composed of initialization registration, access query, access record, key update, node leave, and revocation. The core functionality for verifying the access query transaction is based on logical topology, which is shared by all SAMs in the same deployment domain and is changed by these operations. The secret access key $k_s$ is stored in the UE, and the ciphertext of $k_s$ encrypted by key encryption key $kek$, denoted as $EN_{kek}(k_s)$, is stored in the blockchain operated by the SAMs for audit purposes.

#### *a: INITIALIZATION REGISTRATION*

The system initializer chooses the hash algorithm, the asymmetric cryptographic algorithm, the symmetric cryptographic algorithm and the consensus algorithm to construct the blockchain-based distributed key management system. The system initializer also publishes the related parameters. Then,

① Each UE selects its private key $pv$ and generates the public key $pb$. The UE randomly generates secret access key $k_s$ for secure access and key encryption key $kek$.

② Each UE packages encrypted secret access key $EN_{kek}(k_s)$, key version $kv$, key lifetime $kl$, timestamp $ts$, parent information tuple $PIT = (\langle IMEI\rangle, \langle p, b\rangle)$, deployment tuple $DT = (ad, ld)$, authorization assignment mode $AM$, and the public key of its own $pb$ into a transaction and then signs and broadcasts the transaction to the SAMs, where IMEI is international mobile equipment identity, $ad$ is the application domain and $ld$ is the location domain of the UE.

$$initialization = Sig\left\{\begin{matrix} EN_{kek}(k_s), kv, kl, ts, \\ PIT, DT, AM, pb \end{matrix}\right\}_{pv} \quad (1)$$

③ Each SAM collects the transactions of the UEs that belong to it in a specified collection period and propagates all transactions to other SAMs in the same deployment domain at the end of the collection procedure. After the propagation procedure, all SAMs have the same transaction queue, and it is time to mine the first block of their blockchain.

④ The first block is mined when a SAM solves the POW at a certain difficulty by finding the target number of zeros $n_{zeros}$ at the start of the hash result of the block header. Once the block has been mined and transactions have been recorded, the logical topology of the network has been built and a consensus has been reached for all SAMs.

#### *b: ACCESS QUERY AND ACCESS RECORD*

① When a subject UE-A needs to access an object UE-B in the same blockchain, it launches an access query transaction to the SAM to obtain access permission:

$$query_{(A,B)} = Sig\{UIT_A, UIT_B, ts\}_{pv_A} \quad (2)$$

where $UIT_x = \left(IMEI_x, pb_x\right)$ is the UE information tuple.

② The obligated SAM-1 collects this transaction and propagates it to other SAMs. After the transaction is verified according to the logical topology shared by all SAMs and recorded into a new block by SAM-i, SAM-1 sends a license back to UE-A. Notably, the license signed by SAM-i is not a transaction and is not recorded into the block.

$$licence_{(A,B)} = Sig\left\{UIT_A, UIT_B, ts, kv, kl, accperm\right\}_{pv_{SAM-i}} \quad (3)$$

③ When UE-A receives the license from SAM-1, UE-A decides when or whether to access UE-B during the key lifetime. UE-A sends $licence_{(A,B)}$ to UE-B. After UE-B verifies SAM-i's signature, UE-B sends back the valid $kek$ encrypted by $pb_A$ and launches a new transaction to the SAM to record the access behavior.

$$decryption_{(A,B)} = Sig\left\{UIT_A, UIT_B, EN_{pb_A}(kek), ts\right\}_{pv_B} \quad (4)$$

$$accrecd_{(A,B)} = Sig\left\{license_{(A,B)}, ts\right\}_{pv_B} \quad (5)$$

### *c: KEY UPDATE*

For safety reasons and, for example, because of key expiration for past authorized nodes, the UE needs to periodically update the access keys; therefore, the UE must send a key update transaction to a SAM to record the new key and its lifetime.

$$keyupdate_{(A)} = Sig\left\{\begin{array}{l} EN_{kek_{new}}\left(k_{s_{new}}\right) \\ , kv_{new}, kl_{new}, ts \end{array}\right\}_{pv_A} \quad (6)$$

### *d: LEAVE*

Before a node leaves the network, it generates a new transaction to declare incoming actions and indicates its own parents to be its children's new parent nodes. This operation is optional, and each node decides whether to accept this arrangement according to the authorization assignment mode. This optional relation assignment will be explained in Section III. B (4).

$$leave_{(A)} = Sig\left\{[relation\_assignment]_{optional}, ts\right\}_{pv_A} \quad (7)$$

### *e: ADVERSARY REVOCATION*

If a node is discovered to be compromised or malicious, the SAM signs a new transaction and records it into a new block to declare the revocation.

$$revocation = Sig\left\{[revocation\_list], ts\right\}_{pv_{SAM-i}} \quad (8)$$

The access query and access record procedures are illustrated in Figure 4.

### 2) TRANSACTION FORMAT

In the proposed scheme, transactions are designed to encapsulate all system operations transferred from the UE to the SAM and vice versa. As shown in Table 1, four fields are contained in the universal transaction header to merge all these operations into a universal transaction format, and all other fields are optional according to the operation.

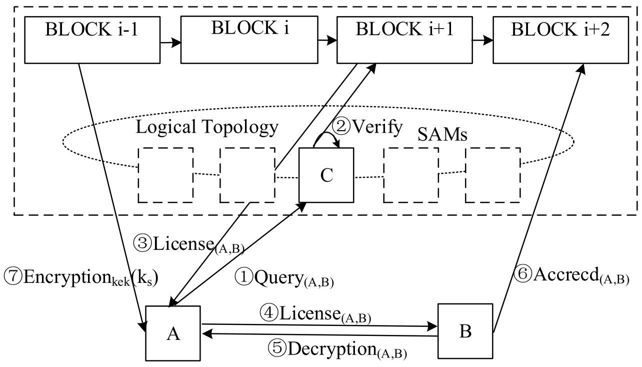


**FIGURE 4.** Access query and access record procedure of BDKMA.

**TABLE 1.** Format of transaction.

| **Transaction Header** | | | | | | |
|---|---|---|---|---|---|---|
| Hash of the transaction | | | | | | |
| Hash of the previous transaction | | | | | | |
| Signature of the transaction signed by UE or SAM | | | | | | |
| Operation type | | | | | | |
| **Payload** | Operation type | | | | | |
| | Initialization | Query | Record | Update | Leave | Revocation |
| *pb* | √ | / | / | / | / | / |
| *kv* | √ | / | √ | √ | / | / |
| *kl* | √ | / | √ | √ | / | / |
| *ts* | √ | √ | √ | √ | √ | √ |
| *PIT* | √ | / | / | / | / | / |
| *UIT* | | √ | √ | / | / | / |
| *DT* | √ | / | / | / | / | / |
| *AM* | √ | / | / | / | / | / |
| $EN_{kek}(k_s)$ | √ | / | / | √ | / | / |
| *Accperm* | / | / | √ | / | / | / |
| *Relation Assignment* | / | / | / | / | √ | / |
| *Revocation List* | / | / | / | / | / | √ |

The hash of the previous transaction in the transaction header is the identification of the pervious operation transaction of the UE and is used to bind and rapidly position all operations throughout the full lifecycle.

The payload of the transaction is indicated by the operation type field in the header; different contents are included in the payload for each operation type. The public key of the UE is assigned at the initialization registration stage, as are the $PIT$, $DT$, and $AM$. To maintain the confidentiality of $k_s$ and to provide an opportunity to audit when a dispute arises, $EN_{kek}(k_s)$ is stored in the blockchain. As a result of key assignment and reassignment, the key version, key lifetime and encrypted secret access key are updated to the latest versions. The timestamp is used to indicate the generation time of the transaction and to compare it with the key lifetime to distinguish whether the access query is authorized. The hash value of the transaction is signed by the UE or SAM with its private key to ensure that a malicious user cannot forge a valid transaction.

Notably, not all messages must be recorded into the blockchain. The license message is transmitted between the

**TABLE 2. Format of block.**

| Block Header | |
|---|---|
| Field | Description |
| Version | Block version |
| Previous Block Hash | Hash of the previous block |
| Merkle Tree Root | Hash of the Merkle tree root |
| Timestamp | Generation time of this block |
| Target Difficulty | The target POW difficulty |
| Nonce | A solution for the POW proof |
| Signature | Signature of the SAM |
| **Block Payload (Transactions)** | |
| Transaction 1,2,… | |

UE and SAM when an access query is authorized and between UEs when an actual access action occurs. A decryption message is transmitted between UEs to securely hand over the key encryption key *kek*. Neither of these types of message is recorded into the block.

### 3) BLOCK FORMAT

In contrast to the diverse format of the transactions, the block header, which contains seven fields, is formalized (Table 2), similar to the block in [28] and [35]. At the end of the propagation procedure, all SAMs have the same transaction queue, and it is time to mine the next block. All transactions are merged into the Merkle [36] tree, which provides an efficient way to ensure the integrity of the transactions. The target difficulty is the number of zeros $n_{zeros}$ at the start of the hash result of the block header when a valid solution has been found, which means a SAM has completed the POW proof, and the solution will be filled into the nonce field.

### 4) REINFORCE SCALABILITY AND EXTENSIBILITY

We have presented the method for constructing and maintaining the multiblockchain in the cloud to support cross-domain interaction in Section III. A. With the advantages of the multiblockchain, we can enhance scalability by reducing the size of each block in every deployment domain, accelerating the collection, propagation, and mining procedures and saving the storage space of the SAMs. The following two methods are used to enhance the extensibility of our proposed scheme.

#### *a: DIFFERENT AUTHORIZATION ASSIGNMENT MODES*

As mentioned in the initialization registration section, UEs declare their parent information tuple *PIT* individually; it is difficult to enumerate every parent of a UE, as it does not know more information about the network topology. To solve this problem, the UE can declare *PIT* with the help of a trusted node, i.e., its own access manager. Because the IoT system, which prefers to keep the user's key information secure and out of the KGC's control, includes UEs ranging from high security requirements to low security requirements, different strategies are required for different UEs.

#### *(i) PRIVATE ASSIGNMENT MODE*

For UEs with high security requirements, assigning parents by other nodes is unacceptable. In such cases, e.g., vehicles in vehicular communication systems (VCSs) and intelligent locks of smart homes, the node itself must assign the nodes that can access it.

#### *(ii) PROTECTED ASSIGNMENT MODE*

For UE with general security requirements, parents can be assigned by the access manager. In this mode, UE not only admits the parents assigned by itself but also admits the parents assigned by its access manager. In this case, the UE must assign its access manager(s) during the initialization registration stage. An example is personal healthcare devices, such as smart bracelets, where the user's cellphone is the access manager. Personal laptops and clinical history storage devices can be assigned authorization to access the encrypted information in the smart bracelet without awareness.

#### *(iii) PUBLIC ASSIGNMENT MODE*

In cases of low security requirements, the UE usually takes on the role of monitoring objects in the surrounding environment and is therefore accessed by many nodes. In this mode, the UE can be extensively accessed by the parents assigned by itself and the ancestors of its parents.

Before UE leaves the network, its children in the protected or public assignment mode may need to be reassigned new parent(s). The UE generates relation assignment information to tell the SAM to change the related topology so that its own parents become its children's new parent nodes.

We simplified the *PIT* assignment procedure and enhanced the network extensibility by means of these three authorization assignment modes.

#### *b: GROUP ACCESS PATTERN*

When a subject node needs to obtain many UEs' access keys, the trivial method is for the node to launch access queries for each request, which is inefficient. To improve the performance in this case, we introduce the group access pattern strategy. Compared to the $licence_{(A,B)}{=}Sig\{UIT_A, UIT_B, ts, kv, kl, accperm\}_{pv_{SAM-i}}$ used in the normal method, we issue a weightier license to UE-A by replacing the $UIT_B$ of each object in $GA = \{UIT_{object}\}$ with several coparents of $GA$. Because the $kv, kl$ of each object is different, no more information is required, and the new weightier license has the following format:

$$licence_{(A,GA)}{=}Sig\left\{\begin{array}{l} UIT_A, [coparents], \\ ts, accperm \end{array}\right\}_{pv_{SAM-i}} \quad (9)$$

The authorization assignment modes and group access pattern are depicted in Figure 5. At the initialization registration stage, UE-9 assigned its access manager as UE-10 in a direct way. Afterwards, UE-10 assigned UE-2 and UE-4 to be authorized to access UE-9 in public or protected assignment mode. Therefore, UE-2,4,10 are permitted to access UE-9.

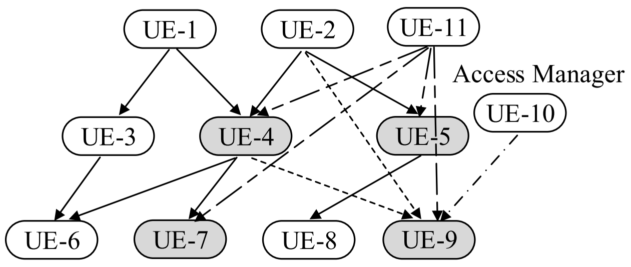


**FIGURE 5.** Using authorization assignment mode and group access pattern to achieve hierarchical access control.

**TABLE 3.** Time elements of the processing procedures.

| Notation | Description |
|---|---|
| $t_U^s$ | Processing time to sign messages for UE |
| $t_U^v$ | Processing time to verify signatures for UE |
| $t_S^s$ | Processing time to sign messages for SAM |
| $t_S^v$ | Processing time to verify signatures for SAM |
| $t_p$ | Propagation time for transactions of each SAM |
| $t_m$ | Block mining time |
| $t_c$ | Transaction collection time |
| $t_{EA}$ | Processing time to encrypt messages (asymmetric) |
| $t_{DA}$ | Processing time to decrypt messages (asymmetric) |
| $t_{DS}$ | Processing time to decrypt messages (symmetric) |

If an authorized UE-11 wishes to access UE-4,5,7,9 and sends an access query to SAM, a signed license with coparents {UE-4,5} will send back to it for group access. Hence, convenient hierarchical access control is achieved since a weightier license is used for the subject to access multiple objects.

### 5) TIME COMPOSITION

Table 3 shows all time elements of the distributed key management scheme. To simplify the time composition model, we ignore the negligible propagation time of the transactions between a UE and the neighboring SAM and the look-up time of the SAM. The subject sends an access query to the neighboring SAM to obtain a license to access the target object, and the processing time for this situation is as follows.

$$t_{QUERY} = t_U^s + t_S^v + t_p + t_m + t_S^s \tag{10}$$

Considering that SAMs propagate their transactions only at the end of the collection procedure, the above $t_{QUERY}$ is the minimum time to obtain the license when the query transaction is the last transaction in the collection period. By contrast, the maximum time $t'_{QUERY}$ to obtain the license occurs when the query transaction is the first transaction in the collection period. Therefore, we derive the average time consumption $\tilde{t}_{QUERY}$ to obtain the license for uniformly distributed incoming transactions.

$$t'_{QUERY} = t_U^s + t_c + t_p + t_m + t_S^s \tag{11}$$

$$\tilde{t}_{QUERY} = t_U^s + \frac{t_c + t_S^v}{2} + t_p + t_m + t_S^s \tag{12}$$

The processing time for other operations is similar to that of the access query; the only difference is that a message is sent back to the UE.

When UE-A obtains a license to access UE-B, it sends an access message to the target. Because the license identifies UE-A, UE-B must sign only the decryption message. Since UE-B needs to verify the signature of SAM-i and UE-A needs to verify the signature of UE-B, $2t_U^v$ is consumed. The encryption key is encrypted by UE-A's public key and is then used to decrypt the encrypted access key recorded in the blockchain. Therefore, the time composition of the access procedure is as follows.

$$t_{ACCESS} = t_U^s + 2t_U^v + t_{EA} + t_{DA} + t_{DS} \tag{13}$$

When the transaction is sent out from the UE after the signature procedure, the transaction is verified and recorded into the next block. The average processing time for certain transactions is the same as $\tilde{t}_{QUERY}$. Let $n_T$ be the number of transactions launched within collection period $t_c$; i.e., $n_T$ transactions are sent to the SAMs during the collection period and wait to be recorded. The overall time consumption for UEs and SAMs is as follows:

$$\tilde{T}_{process} = n_T \times t_U^s + t_c + t_p + t_m + t_S^s \tag{14}$$

Therefore, the average time consumption for each transaction is:

$$\tilde{t}_{process} = \frac{1}{n_T}\tilde{T}_{process} = t_U^s + \frac{1}{n_T}\left(t_c + t_p + t_m + t_S^s\right) \tag{15}$$

The cross-domain operation is authenticated when the verification results from the cloud manager are sent back to the SAM and are recorded into the block. Because of the abundant resources of the cloud managers, we ignore the verification processing time. The average time consumption for the cross-domain operation is related to $t_c$, $t_p$ and the ratio of cross-domain transactions to all transactions $p_{CD} = \frac{n_{crossdomaintrans}}{n_T}$, $0 \leq p_{CD} \leq 1$.

In the case of $t_c \geq 2t_p \times p_{CD}$, the verification results are expected to be returned before the next propagation period of the SAMs; therefore,

$$\begin{aligned}\tilde{t}_{CD} &= t_U^s + \frac{t_c + t_S^v}{2} + t_c + t_p + t_m + t_S^s \\ &= t_U^s + \frac{t_S^v}{2} + t_S^s + \frac{5}{2}t_c\end{aligned} \tag{16}$$

By contrast, in the case of $t_c < 2t_p \times p_{CD}$, the transaction is recorded into the next block when returned to the SAM.

$$\begin{aligned}\tilde{t}_{CD} &= t_U^s + \frac{t_c + t_S^v}{2} + \left\lceil \frac{2t_p \times p_{CD}}{t_c} \right\rceil \times t_c + t_p + t_m + t_S^s \\ &= t_U^s + \frac{t_S^v}{2} + t_S^s + \frac{3 + 2\left\lceil \frac{2t_p \times p_{CD}}{t_c} \right\rceil}{2} t_c\end{aligned} \tag{17}$$

### 6) DYNAMIC TRANSACTION COLLECTION TIME

To accelerate the mining procedure, a reasonable target difficulty, which influences the block mining time, must be considered, along with the collection time, the propagation time and the number of transactions in each block. With the determined access query response time, which has a direct bearing on system performance, a dynamic target difficulty is needed to adjust the mining time.

The operating cycle is depicted in Figure 6. In the collection period, each SAM collects transactions from its management domain and then propagates all transactions to other SAMs. At the end of the propagation period, every SAM begins to mine the new block that contains the same transaction queue. When the new block is found by SAM-i, this cycle of block mining is complete.

We further studied the time composition of the operating cycle to eliminate the unoccupied time among these three periods. Let $n_S$ be the number of SAMs in the network; the average number of transactions for each SAM is $\frac{n_T}{n_S}$. The average length of the transaction is denoted as $l$ bits; therefore, the total length of the transactions for each SAM is $\frac{n_T}{n_S} \times l$. We denote the transmission rate as $s$ bps; then, the expected average propagation time is $\tilde{t}_p = \frac{n_T \times l}{n_S \times s}$. As shown in Figure 6, the following equation should be satisfied to eliminate the unoccupied time among these three periods:

$$t_c = \tilde{t}_p + \tilde{t}_m \tag{18}$$

where $\tilde{t}_m$ is the expected average mining time for a specified target difficulty.

Let $C_{hash}$ be the average hash capacity of a SAM; then, we have the following equation.

$$\left(\tilde{t}_m \times C_{hash}\right) \times \frac{1}{2^{n_{zero}}} = 1 \tag{19}$$

The target difficulty $n_{zero}$ can be represented as

$$\begin{aligned} n_{zero} &= log_2\left(\tilde{t}_m \times C_{hash}\right) \\ &= log_2\left[\left(t_c - \frac{n_T \times l}{n_S \times s}\right) \times C_{hash}\right] \end{aligned} \tag{20}$$

With the help of equation (20), we can dynamically change the target difficulty $n_{zero}$ while the collection period $t_c$ is set and the network status fluctuates with time.

## IV. MODEL ANALYSIS

### A. PERFORMANCE EVALUATION

The performance evaluation of the blockchain-based distributed key management scheme for hierarchical access control was conducted via simulation. The performance evaluation consists of three parts. First, we state the assumptions we used in the simulation. Second, we study the interrelationships of parameters that influence system performance. Finally, we assess the performance in the cross-domain access case and compare the transaction processing time against different transaction collection times.

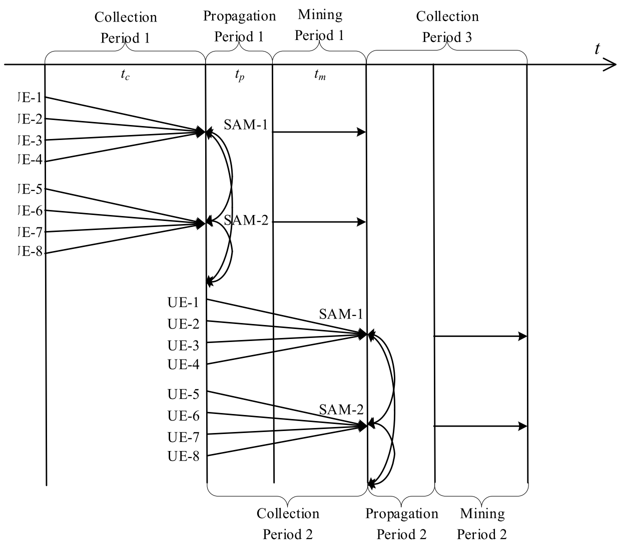


**FIGURE 6. Time composition of the processing procedure.**

#### 1) SIMULATION ASSUMPTIONS

Our results are obtained using OMNeT++ 5.4.1 [37], [38] with ECIES [39] and ECDSA using elliptic curve secp160r1 [40] in Crypto++ [41] to encrypt the secret access keys $k_s$ and to sign the transactions. We used AES as the block cipher algorithm to encrypt $k_s$ by $kek$, which is stored in the blockchain. We assume that the average length of the transaction is 256 bytes, including $160 \times 2$ bits for signature, $256 \times 2$ bits for two hash values, and 152 bytes for the other fields. We conducted the simulations on a laptop with an Intel Core i5 CPU and 8 GB RAM. The performance of the simulations is related to the transaction collection time and the number of SAMs and UEs. Therefore, the simulations are based on the following assumptions.

(1) The total number of UEs $N_{UE}$ in one blockchain (deployment domain) is the sum of UEs in the different management domains managed by SAMs. The management domains have the same number of UEs $n_u$; therefore, $N_{UE} = n_u \times n_S$, where $n_S$ is the number of SAMs in one blockchain.

(2) The power of a SAM is positively related to the number of UEs in its management domain, that is, SAMs have the same transmittability to propagate transactions and the same mining time to find the next block.

(3) The initialization of UEs and the access query transaction occurrence rate follow an exponential distribution:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{21}$$

where $\mu = \frac{1}{\lambda}$ is the mean.

(4) As mentioned in the dynamic transaction collection time section in Section III. B, the mining procedure is activated at the end of the propagation procedure. According to assumptions (2) and (3), each SAM has the same transaction queue, which guarantees the consistency of the new block.

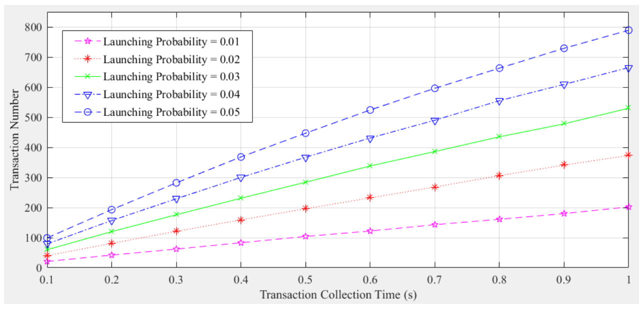


**FIGURE 7. Transaction number with respect to the transaction collection time under different launching probabilities.**

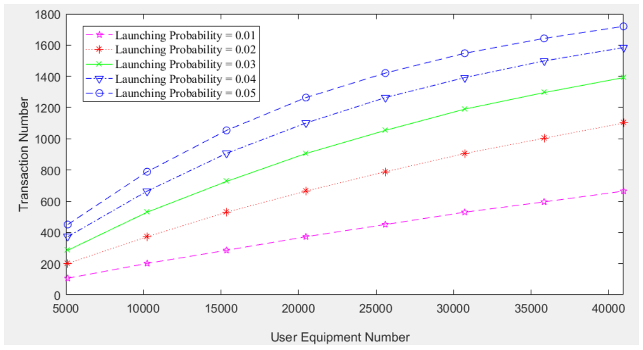


**FIGURE 8. Transaction number with respect to user equipment number under different launching probabilities.**

## 2) PARAMETER INTERRELATIONSHIP STUDY

To provide a low-latency key management function for hierarchical access control, multi-SAMs are used in our proposed scheme. The size of the block is influenced by the transaction collection time $t_c$, the total number of UEs $N_{UE}$ and the probability (interval) of launching an access query transaction. In this section, we study the interrelationship of the above parameters.

Figure 7 shows the average number of transactions $n_T$ in terms of the transaction collection time $t_c$. We simulated the performance of different launching probabilities at fixed numbers of UEs $N_{UE} = 10240$ and SAMs $n_S = 64$. The average number of transactions increases as the transaction collection time increases. Additionally, $n_T$ increases nearly linearly with $t_c$ if the UEs have a low probability of launching a transaction. However, the rate of increase decreases as the probability of the UEs launching a transaction increases. The nonlinearity is caused by the autonomous decision of subjects of when or whether to access the objects in the key lifetime. The average number of transactions in terms of launching probability with a fixed $t_c = 1s$ and different $N_{UE}$ is shown in Figure 8. At a low launching probability $p = 0.01$, $n_T$ increases from 107 when $N_{UE}= 5120$ to 665 when $N_{UE}= 40960$. At a high launching probability $p = 0.05$, $n_T$ increases from 451 when $N_{UE}= 5120$ to 1720 when $N_{UE}= 40960$. The simulation results demonstrate the scalability of the proposed scheme.

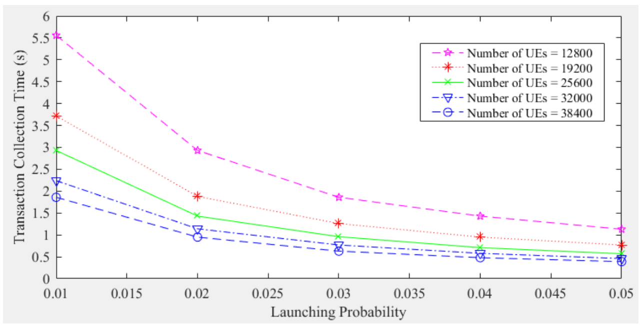


**FIGURE 9. Transaction collection time with respect to launching probability under different numbers of user equipment.**

## 3) BLOCKCHAIN PERFORMANCE EVALUATION

To implement our scheme for a real network, a relatively stable block size is needed to obtain the target difficulty for all SAMs. Figure 9 plots the transaction collection time $t_c$ in terms of the launching probability. As noted previously, the average transaction length is 256 bytes, and we limited the average size of a block to 256 KB, with 1024 transactions. When $N_{UE} = 12800$, the weighted average of the transaction collection time is $t_c = 1.89s$, and in the larger network where $N_{UE} = 38400$, the weighted average of the transaction collection time is $t_c = 0.63s$. Notably, the network size indicates only the size of the blockchain network in one deployment domain, which frequently provides interactions between UEs. For the larger case with 240 SAMs, each comprising 160 UEs, there are 4.3 transactions in each SAM in a transaction collection period on average. Therefore, 8.4 *ms* is consumed to broadcast transactions to other SAMs over 1 Mbps transmission line. The expected average mining time is set to $\tilde{t}_m = 621.6\ ms$ to eliminate the unoccupied timẽ. For a SAM device with a hash capacity $C_{hash} = 250\ K$, 155.4 $K$ hashes on average are needed to mine the next block. Therefore, the target difficulty is $log_2\,(1554000) \approx 17$ bits, i.e., 17 continuous zeros should be found at the start of the hash result of the block header.

In the time composition Section III. B, we defined the overall time consumption $\tilde{T}_{process}$ and the average time consumption $\tilde{t}_{process}$ for each transaction in a block. In our analysis, we use 0.87 *ms* for $t_S^v$, 0.47 *ms* for $t_S^s$ and 3.29 *ms* for $t_U^s$. Therefore, we obtain the time consumption in terms of $n_T$ for the proposed scheme, as illustrated in Figure 10, where $t_c = 1\ s$. With sufficient power to handle the verifications, i.e., $\frac{n_T}{n_S} \times t_S^v \leq t_c$, the average time consumption for each transaction decreases with the transaction collection time because the signing time is the same for different $n_T$. The average processing time $\tilde{t}_{QUERY}$ for certain transactions is constant in terms of $n_T$ because the UE receives the access license only until the next block is mined.

Figure 11 plots the numbers of three different types of transactions in terms of processing time. In this simulation, we employ 4096 UEs with $t_c = 0.5\ s$. The probability that a UE initializes or launches a transaction follows an exponential distribution with $p = 0.1$ ($\mu = 10\ s$).

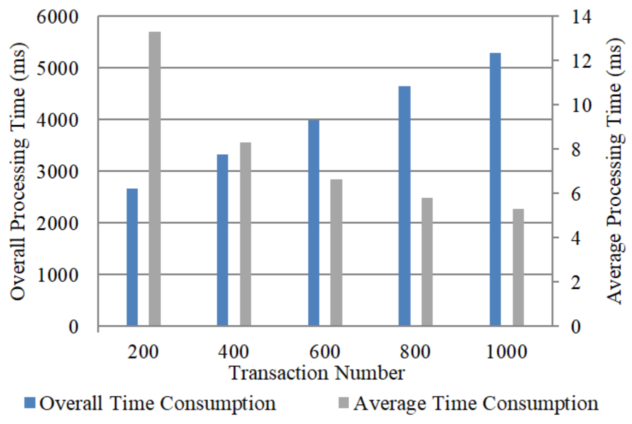


**FIGURE 10. Overall time consumption and average time consumption with respect to the number of transactions in each block.**

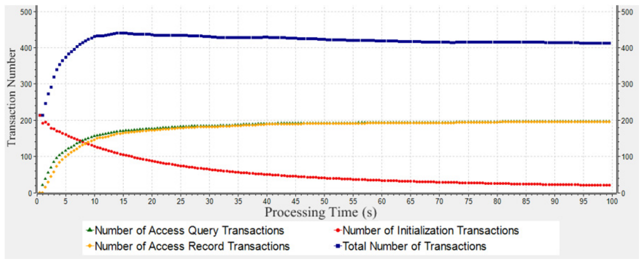


**FIGURE 11. Average numbers of different types of transactions during the processing time.**

The total number of transactions increases with the processing time from system initialization to a relatively steady state when all UEs have joined the network and process the normal access query and record operations. Moreover, the difference between the number of access queries and access record transactions is caused by the autonomous decision of subjects of when or whether to access the objects in the key lifetime.

### 4) CROSS-DOMAIN OPERATION PERFORMANCE EVALUATION

We introduced several blockchains for different deployments and employed cloud managers to operate the multi-blockchains in order to reduce the number of transactions in one block, decrease unnecessary transactions for a group of UEs and reduce the verification time for SAMs. When cross-domain operation occurs, cloud managers judge and verify requests and return the results back to both sides of the participating blockchain. The cross-domain operation is permitted when the verification results from the cloud manager are sent back to the SAM and recorded into the block.

Figure 12 depicts the performance of the cross-domain operations. Because the time consumption of SAMs receiving the verification results in the cross-domain situation are related to $p_{CD}$ and $t_p$, we simulated the key transmission time for different transaction collection times with respect to $P_{CD} \times t_p$. In the case $N_{UE} = 10240$ and $p = 0.01$, the key transmission time increases every time $P_{CD} \times t_p$ exceeds half of $t_c$. Because all transactions are returned to the SAMs in the

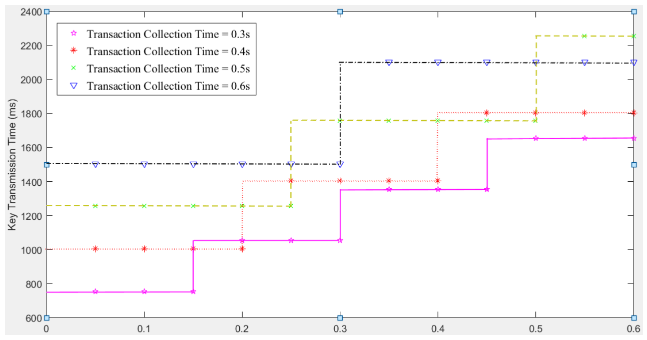


**FIGURE 12. Key transmission time with respect to propagation time and the ratio of cross-domain transactions to all transactions under different transaction collection times.**

same propagation period, the time consumption of the cross-domain operations is the same during each $\frac{1}{2}t_c$ period.

## B. COMPARISON OF SCHEMES TO CONVENTIONAL STRATEGIES

Because of the requirements of the IoT and the advantages of introducing the blockchain and fog computing into the IoT mentioned above, we believe that combining these two technologies will lead to good effects. In this section, we compare the advantages of our blockchain-based distributed key management scheme with the other KMS mentioned above.

### 1) DECENTRALIZATION

A novel blockchain concept is introduced into the proposed scheme to eliminate potential challenges, such as abuse and information leakage. The SAMs act as the CA or KGC in conventional centralized schemes and authorize access queries in a public and decentralized manner. Trustworthiness exists in the consistency maintained by the consensus algorithm and reinforced by subsequent blocks. At the end of the system initialization stage, a common logical topology takes shape among the SAMs to help judge the rights of a certain query. Moreover, different authorization assignment modes and group access patterns are used to achieve hierarchical access control because a weightier license is returned to authorized subjects to indicate greater privilege. Therefore, the advantage of eliminating the trusted third party makes the proposed scheme suitable for user privacy-oriented scenarios.

### 2) AUDITABILITY

Nonrepudiation is an important feature for maintaining the stability and reliability of a system in an open network environment. Debates arise when malicious participants become involved in the key management procedure and can be divided into several categories:

(1) unauthorized access;

(2) key management operations (update, topology modification) launched by unauthorized entities;

(3) denial of legitimate requests; and

(4) invalid key encryption key offered by an object.

**TABLE 4.** Scheme comparison.

| | Hierarchical Schemes | Preshared Key Schemes | Mathematical Framework | Key Pool Framework | PKC | Our Scheme |
|---|---|---|---|---|---|---|
| Decentralization | NO | NO | NO | NO | NO | YES |
| Auditability | Untrustworthy | Untrustworthy | Untrustworthy | Untrustworthy | YES – based on trusted CA | YES – based on consensus |
| Scalability | Low | Low | High | High | Medium | Medium |
| Extensibility | Low | Low | Medium | Medium | High | High |
| Key Update Complexity | Medium | High | High | Negligible | Medium | Low |
| Central Point Overhead | Medium | Medium | Medium | Medium | High | None |
| Communication and Computational Overhead | High | Low | Low | Low | High | High |
| Resilience | Medium | Low | Medium | Low | High | High |

Compared with previously reported schemes [5]–[8], [10]–[12], these schemes do not have the ability to determine the source of operations or to store trusted historical records to provide auditability. Both the proposed scheme and [9] provide feasible and fine-grained auditability by digital signature technology. The difference is that PKI pushes the responsibility to the trustworthiness of third parties, while the proposed scheme stores all key management operations into the blockchain to achieve full lifecycle auditability.

### 3) SCALABILITY

This feature represents the amount of data that the UEs must store to implement the key management function. The schemes proposed in [6] and [8] have excellent scalability due to the distribution method and optimized key space structure. The preshared key schemes require more space to store keys when the network increases to a large number of users; therefore, the performance is normal. For HKAS [10]–[12], the UEs need to store only a set of cryptographic materials to derive the access keys, but the size of the materials is significantly increased due to asymmetric cryptography. The scalability of the PKI is not great but is sufficient because only the certificates of the CA need to be stored.

In our scheme, a UE needs to store only self-correlative information (i.e., its own asymmetric key pair and key encryption key). No additional keys and certificates are needed because the authentication is based on the logical topology maintained by the SAMs. Meanwhile, the SAMs store only the information of their own deployment domain, and the complete multiblockchains are stored in the cloud. All these advantages result in good performance.

### 4) EXTENSIBILITY

In the context of IoT, a large number of devices gradually join the network, and extensibility represents the capability and convenience of increasing the network scale. Schemes using a key pool strategy [6] assign a key subset from the key pool for each node, and the key pool size is adjusted based on the network size. The number of columns of the public Vandermonde matrix G in scheme [11] determines the upper bound of the network size. These two approaches have stable and static extensibility because the upper bound of the network size is determined before deployment. The preshared key scheme is not extensible because the key size increases exponentially and cannot provide backward access capability. HKAS can be extended to a large number of nodes but is not suitable for cases in which the topology changes frequently because the access keys are derived by predefined cryptographic materials. PKI and the proposed scheme can conveniently be extended to a large scale. In the proposed approach, frequent access operations are locked in a single blockchain within the same deployment domain, and few cross-domain accesses are provided by the cloud manager.

### 5) KEY UPDATE COMPLEXITY

In the preshared key scheme and Blom's scheme, the key update operation results in computation and communication overhead for a large number of other nodes and cannot be performed in an asynchronous pattern; that is, periodic global-scale key reassignment is required for KGC. The key update in hierarchical KMS leads to changes in key pairs for relevant nodes. The involved nodes are significantly reduced because of the key derivation pattern and organization form. For key pool schemes, the updated node needs only to calculate and find a new key chain. However, the overhead to find the path is pushed to others, which causes a heavy burden. For PKI strategies, each UE needs to update its certificate individually, and it is friendly to other nodes. For the proposed scheme, the update operations are performed asynchronously, the new access key encrypted by the new key encryption key is recorded and stored in the blockchain, and a new access query procedure is launched after key expiration.

#### 6) CENTRAL POINT OVERHEAD

For strategies with KGCs, the central point overhead is determined mainly by two factors: the number of nodes involved in the key update and the complexity of each operation. For schemes [5]–[8], KGC must perform symmetric key updates for a large number of nodes, and for HKAS [10]–[12], KGC must perform asymmetric key updates for partial nodes. The CA must generate and distribute certificates for all nodes, which results in substantial overhead. The proposed scheme eliminates the third party, and the UEs and SAMs play the role of the central point and apportion the overhead.

#### 7) COMMUNICATION AND COMPUTATIONAL OVERHEAD

Schemes using a preshared key strategy and depending on the mathematical key structure have advantages in terms of computational and communication overhead. Conversely, the asymmetric key strategies have large overhead.

#### 8) RESILIENCE

The key management system for IoT must maintain service without interruption to adapt to dynamic changes in a large number of cases. In the proposed scheme, SAMs maintain and manage the blockchain in a distributed manner, thereby avoiding the single point of failure suffered by other schemes. Trustworthiness exists in the consistency maintained by the consensus algorithm and is reinforced by subsequent blocks that cause strong resilience.

The scheme comparison from different measurements is shown in Table 4. It can be seen that the most signifacant advantage of the proposed scheme is that reliable auditability is achieved due to decentralized deployment and management. Although the communication and computational overhead for the entire network are at a disadvantage, the overhead of certain UEs and SAMs is low. Therefore, the proposed scheme is suitable for privacy-oriented IoT scenarios.

## V. CONCLUSION

In this paper, we proposed a novel blockchain-based distributed key management architecture (BDKMA). A novel blockchain concept with cloud computing and fog computing is introduced to satisfy the decentralization, fine-grained auditability, high scalability and extensibility requirements, as well as the privacy-preserving principles for hierarchical access control in IoT. We split the network into different side blockchains on the basis of the deployment domain to accelerate verification and save valuable storage space for IoT devices. The side blockchains are maintained by the SAMs in each domain, and multiblockchains are stored in the cloud to support cross-domain interaction. The simulation results show that the multiblockchain structure substantially improves system performance, and the scalability is excellent as the network grows. Furthermore, the dynamic transaction collection time adjustment enables the performance and system capacity to be optimized for various environments.

In the future, we will explore and design a feedback mechanism for SAMs and cloud managers to facilitate the persistency of the blockchain-based IoT ecosphere.

## REFERENCES

[1] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the Internet of Things," *Comput. Elect. Eng.*, vol. 37, no. 2, pp. 147–159, Mar. 2011.

[2] *IoT Developer Survey*. (2017). [Online]. Available: https://www.slideshare.net/ianskerrett/iot-developer-survey-2017

[3] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, "World of empowered IoT users," in *Proc. IEEE 1st Int. Conf. Internet-Things Design Implement. (IoTDI)*, Berlin, Germany, Apr. 2016, pp. 13–24.

[4] S. Nakamoto. (2018). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/en/bitcoin-paper

[5] W. Bechkit, Y. Challal, A. Bouabdallah, and V. Tarokh, "A highly scalable key pre-distribution scheme for wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 2, pp. 948–959, Feb. 2013.

[6] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, Washington, DC, USA, 2002, pp. 41–47.

[7] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. Workshop Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, Paris, France, 1985, pp. 335–338.

[8] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 2, pp. 228–258, May 2005.

[9] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, St. Louis, MO, USA, 2008, pp. 245–256.

[10] A. Castiglione *et al.*, "Hierarchical and shared access control," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 850–865, Apr. 2016.

[11] S. Tang, X. Li, X. Huang, Y. Xiang, and L. Xu, "Achieving simple, secure and efficient hierarchical access control in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2325–2331, Jul. 2016.

[12] J. Yan, J. Ma, and H. Liu, "Key hierarchies for hierarchical access control in secure group communications," *Comput. Netw.*, vol. 53, no. 3, pp. 353–364, Feb. 2009.

[13] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, Jan. 2018.

[14] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *IEEE Int. Congr. Big Data (BigData Congress)*, Honolulu, HI, USA, Jun. 2017, pp. 557–564.

[15] L. Wu, X. Du, W. Wang, and B. Lin, "An out-of-band authentication scheme for Internet of Things using blockchain technology," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Maui, HI, USA, 2018, pp. 769–773.

[16] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sensor Ad Hoc Commun. Netw. (SECON)*, Santa Clara, CA, USA, Oct. 2004, pp. 71–80.

[17] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Cambridge, U.K.: O'Reilly Media, 2014.

[18] P. Vasin. (2014). *Blackcoins Proof-of-Stake Protocol V2*. [Online]. Available: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf

[19] Bitshares. *Delegated Proof-of-Stake Consensus*. Accessed: 2009. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus

[20] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Design Implement.*, New Orleans, LA, USA, 1999, pp. 173–186.

[21] (2016). *Antshares Digital Assets for Everyone*. [Online]. Available: https://www.antshares.org

[22] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork After Withholding (FAW) attacks on bitcoin," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, 2017, pp. 195–209.

[23] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography Data Security*. (Lecture Notes in Computer Science), vol. 61, no. 7. Berlin, Germany: Springer, 2018, pp. 95–102.

[24] 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)*, Vienna, Austria, 2016, pp. 25–30.

[25] N. T. Courtois and R. Mercer, "Stealth address and key management techniques in blockchain systems," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, Porto, Portugal, 2017, pp. 559–566.

[26] M. Conoscenti, A. Vetrò, and J. C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," in *Proc. 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Agadir, Morocco, 2016, pp. 1–6.

[27] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, no. 10, pp. 533–546, Oct. 2016.

[28] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1832–1843, Dec. 2017.

[29] A. Dorri, S. S. Kanhere, and R. Jurdak. (Aug. 2016). "Blockchain in Internet of Things: Challenges and solutions." [Online]. Available: https://arxiv.org/abs/1608.05187

[30] E. Karafiloski and A. Mishev, "Blockchain solutions for big data challenges: A literature review," in *Proc. IEEE 17th Int. Conf. Smart Technol. (EUROCON)*, Ohrid, Macedonia, Jul. 2017, pp. 763–768.

[31] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, Sep. 2018.

[32] M. Samaniego and R. Deters, "Using blockchain to push software-defined iot components onto edge hosts," in *Proc. Int. Conf. Big Data Adv. Wireless Technol.*, Blagoevgrad, Bulgaria, 2016, p. 58.

[33] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in IoT," in *Europe and MENA Cooperation Advances in Information and Communication Technologies* (Advances in Intelligent Systems and Computing), Cham, Switzerland: Springer, 2017, pp. 523–533.

[34] O. Novo, "Blockchain meets IoT: An architecture for scalable access management in IoT," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184–1195, Apr. 2018.

[35] A. Lei, C. Ogah, P. Asuquo, H. Cruickshank, and Z. Sun, "A secure key management scheme for heterogeneous secure vehicular communication systems," *ZTE Commun.*, vol. 21, no. 3, pp. 1–11, Jun. 2016.

[36] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany, 1988, pp. 369–378.

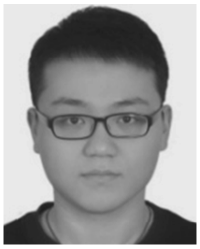
[37] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. Eur. Simulation Multiconf.*, vol. 9, 2001, p. 65.

[38] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[39] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer, 2006.

[40] *Standards for Efficient Cryptography SEC2: Recommended Elliptic Curve Domain Parameters*. Accessed: 2000. [Online]. Available: http://www.secg.org/SEC2-Ver-1.0.pdf

[41] W. Dai. (2009). *Crypto++ library 8.0.* [Online]. Available: http://www.cryptopp.com



**MINGXIN MA** received the B.S. degree in communication engineering from Xidian University, China, in 2014. He is currently pursuing the Ph.D. degree in information security with Xidian University, Xi'an, China, also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, and also with the Beijing Electronic Science and Technology Institute, Beijing. His research interests include key management and the IoT security.

**GUOZHEN SHI** received the M.S. and Ph.D. degrees in computer science from the Beijing University of Posts and Telecommunications, China, in 2008. He is currently an Associate Professor with the Beijing Electronic Science and Technology Institute. His current research interests include network security and embedded systems.

**FENGHUA LI** received the B.S. degree in computer software and the M.S. and Ph.D. degrees in computer systems architecture from Xidian University, China, in 1987, 1990, and 2009, respectively. He is currently a Professor and a Doctoral Supervisor of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences. He is also a Doctoral Supervisor with Xidian University and the University of Chinese Academy of Sciences. His current research interests include network security, system security, privacy computing, and trusted computing.