Computer Science and Information Systems 21(3):807–830 https://doi.org/10.2298/CSIS231002019B

Medical Record Information Storage Scheme based on Blockchain and Attribute Role-Based Access Control

Aoao Bian, Dezhi Han, Mingming Cui, and Dun Li

College of Information Engineering at Shanghai Maritime University
Shanghai China
aobian2022@163.com
dezhihan88@sina.com
mmcui@stu.shmtu.edu.cn
li.dun@telecom-sudparis.eu

Abstract. The integration of the Internet with the healthcare industry has ushered in a transformative revolution in the medical field, where efficient and prompt electronic medical records can save precious resources.However, incomplete or tampered electronic medical records have serious implications on patient. Ensuring the complete, accurate storage of medical record information is an urgent problem that needs to be addressed.Therefore, this paper proposes a blockchain-based solution for secure storage and sharing of medical record information, combining the role-based access control (RBAC) with the attribute-based access control (ABAC).Firstly, by utilizing RBAC and ABAC, dynamic fine-grained access control for medical information is achieved based on role differences.Then, the medical record information is stored in the blockchain through chaincode .In addition, Advanced Encryption Standard and Feature-Aware Stateful Routing technology is applied to further enhance security and storage efficiency of the scheme. Experimental results demonstrate that the proposed solution ensures the security and integrity of medical record, while providing efficient information storage and access.

Keywords: Medical record, blockchain, role-based access control (RBAC), attribute-based access control (ABAC), Feature-Aware Stateful Routing (FASR).

1. Introduction

Medical record information is a valuable treasure.Medical records store all the medical records of patients. These medical records are not only convenient for them to grasp their basic physical conditions, but also for doctors to understand the patient's past medical history, so that they can make further treatment plans according to the actual situation of the patient. With the rapid development of the Internet industry, the medical industry is also moving towards informatization, and medical record information has also been digitized, thus evolving into electronic medical record[1],[2].

Compared with traditional paper medical records, electronic medical records are more convenient to store and query, but the current information storage technology still cannot ensure the security of information, so there are still risks in the transmission process of medical records information. More and more medical institutions use centralized storage to ensure the security of medical record information[3]. However, centralized access storage is easy to cause data theft and loss because it stores data in a central location. If there is a single point of failure in the centralized storage system, all the medical records in the system will be affected, which is likely to lead to the collapse of the medical system, causing immeasurable losses[4]. Another problem with centralized access storage is the inability to share information between different medical institutions. Depending on the hospital, the patient's medical record information is stored in separate storage servers, which are usually only accessible to people with specific permissions internally, and thus do not have any form of interoperability[5]. Most healthcare organizations use security policies to restrict access to and sharing of medical record information. There is a lack of safe and effective medical record information sharing mechanism among institutions[6],[7]. Obviously, this management mode lacks the necessary flexibility and efficiency, which can neither guarantee the security and reliability of medical record information, nor realize the sharing and utilization of medical record information.In fact, medical record information is not only a treasure, but also an important material to understand the patient's physical condition. In order to facilitate the transfer of electronic medical records between systems, most medical institutions adopt different methods to reduce the redundancy of data.

Fortunately, the development of blockchain technology has brought new solutions to address the security storage issue in medical data management[8].However, most of the projects are still in the conceptual stage. To address this issue, according to the characteristics of the information supply chain, this paper uses ABAC and RBAC to divide the access roles in a fine granularity, and combines FASR de-duplication technology to propose a security storage scheme of medical record information based on the blockchain of ABAC, RBAC and Hyperledger Fabric consortium blockchain[9],[10].

The main contributions of this paper are as follows:

1) ABAC, RBAC, and blockchain enables dynamic fine-grained access control for medical information based on role differences. Medical record information is stored in blockchain blocks, and its security is ensured through the formulation of different smart contracts[11].
2) Selecting the appropriate encryption technology to encrypt the medical record information further improves the security of data.
3) FASR deduplication technology is employed to eliminate duplicate medical record information, reducing data redundancy and improving storage efficiency.
4) Simulation experiment results demonstrate that the proposed solution not only ensures the security and data integrity in the processes of medical record storage and sharing, but also improves the efficiency of information storage and access.

The remaining sections of this paper are organized as follows. In Section 2, we summarize the literature related to the program.In Section 3, we provide the necessary background knowledge. In Section 4, we describe the main architecture of the proposed solution and its functional design.And we present the results of the simulation experiments and analyze them in detail in Section 5. Finally, in Section 6, we make concluding remarks on the overall solution and suggest future directions for improvement.

2. Related Work

With the advent of the digital era, the traditional centralized system has been difficult to meet people's needs, and distributed ledger technology, as a decentralized emerging technology, is gradually recognized and applied by people. The emergence of blockchain technology has allowed people to see the future of decentralization and new opportunities in the digital economy. Blockchain is used in many fields and is a trusted data storage technology[12]. Due to its decentralized and tamper-proof characteristics, this technology is often used for information security storage and information traceability. For this reason, a lot of research is currently being done into this technology. In addition, the technology is expanding into more and more applications[13],[14].

Blockchain is essentially a highly scalable distributed database and its tamper-resistant nature can prevent data from being tampered with, deleted or leaked, etc., thus ensuring the security of medical data. Blockchain has been applied in various fields and is a trusted data storage technology[15],[16]. In terms of copyright information, Wang et al.[17]proposed the use of blockchain to prevent the loss of image data. Han et al.[18] designed a blockchain-based copyright certificate storage and transaction system. Jing et al. [19] presented a blockchain-based code copyright management system. Garilli et al.[20] analyzed the pros and cons of using blockchain and smart contracts in a copyright context. Xiao et al.[21] proposed a blockchain-based algorithm for intellectual property copyright protection.

In the supply chain field,Gao and Tan et al.[22,23] explored the application of blockchain technology in logistics supply chain.Musamih et al.[24] achieved efficient product traceability in healthcare supply chains using smart contracts and decentralization. Saberi et al.[25]discussed the potential applications of blockchain technology and smart contracts in supply chain management, providing insights to overcome obstacles in supply chain management.

In the education field, Chen et al.[26] discussed the prospects of blockchain technology in education. Bhaskar et al.[27] summarized the applications of blockchain technology in the education field. Rii et al.[28] constructed a system using blockchain methods to collect information through digital observation and questionnaire surveys.Li et al.[29] proposed a blockchain-based education record storage and sharing solution called EduRSS.

Blockchain has expanded its application scenarios and produced unexpected effects in the medical field. Radanović and Chen et al.[30,31] provided insights into the application of blockchain in the healthcare domain. Li et al.[10] applied blockchain to the supply chain of medical devices and combine RBAC and ABAC for access control management. Qu et al. [32]proposed a blockchain-based medical information system solution. Liu et al.[33] presented a mechanism for sharing medical resources under the conditions of blockchain technology.Bakro et al.[34]analyzed the security aspects of data stored in the cloud through hybrid security system enabled with blockchain technology using cryptographic algorithms ECC and AES.Yassine et al.[35] discussed a new approach that will allow a smart deletion of data stored in the file system as well as its reference in the Blockchain.

However, in the medical field, most of the above projects are still in the conceptual stage. In this paper, we describe the study of this scheme in detail.

3. Preliminaries

In this section, we provide an introduction to several essential concepts, including Hyperledger Fabric, $ABAC$ and $RBAC$ models, AES encryption algorithm, and $FASR$.

3.1. Hyperledger Fabric

Compared to traditional databases, data in a blockchain is stored across multiple nodes in the entire network, with each node holding a complete copy of the ledger[36]. Hyperledger Fabric is a consortium blockchain that allows only authorized users to join the network[37]. Additionally, Hyperledger Fabric utilizes a channel mechanism, where users with the same channel can access each other's account ledgers, thus ensuring the privacy of sensitive information for those users[38]. Fig.1 illustrates the operational architecture of Hyperledger Fabric.

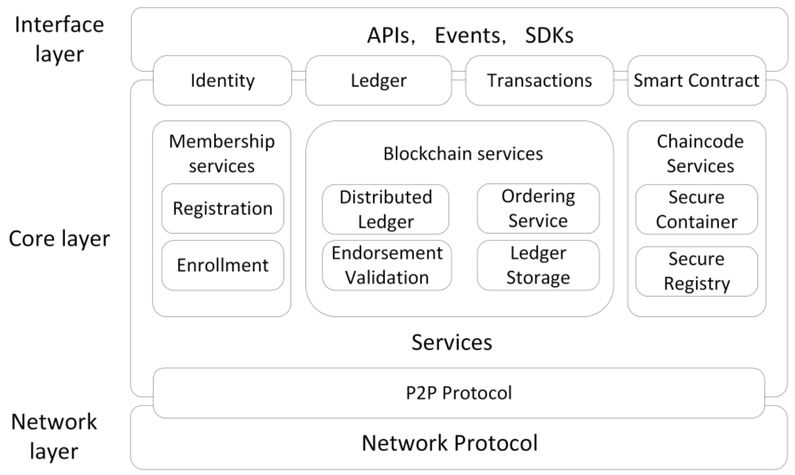


Fig. 1. Hyperledger Fabric's Operational Architecture Diagram

3.2. ABAC and RBAC Models

ABAC

$ABAC$ is an attribute-based access control model that uses attributes to determine whether a user has the right to access resources [39]. Unlike traditional access control models such as Discretionary Access Control (DAC) and Mandatory Access Control (MAC), $ABAC$ emphasizes defining and enforcing policies based on attributes rather than just user identity or resource classification. It utilizes multiple attributes to determine access permissions, making access control more flexible and fine-grained. In healthcare, $ABAC$ can simplify the management of medical records by defining and managing access control policies in a unified location, reducing administrative workload and the chance of errors. In order to achieve dynamic management of medical records, this paper instantiates the $ABAC$ model as an access control policy called Policy, composed of $AS, AO, AP, andAE$. Table 1 below compares $ABAC$ with MAC and DAC.

Table 1. Comparison of ABAC with MAC and DAC Access Control Models

Categories	Flexibility	Security	Extensibility
ABAC	higher	higher	higher
MAC	lower	higher	lower
DAC	higher	lower	lower

RBAC

In $RBAC$, permissions are granted to specific roles and users are then assigned to corresponding roles. This allows for more precise management of access rights, ensuring that users can only access resources within the scope of their job functions. $RBAC$ offers several advantages compared to other access control models. Firstly, $RBAC$ is more flexible and can better adapt to different organizational structures and security requirements. Secondly, $RBAC$ simplifies the process of managing permissions, making it easier and more secure to allocate and modify access rights. Besides, it enhances system security and scalability[40].

3.3. AES

AES is a symmetric encryption algorithm widely used in data protection and security domains[41],[42]. It is currently recognized as one of the most secure encryption algorithms and is also adopted as one of the confidentiality standards by the U.S. government[43].AES encryption and decryption operations are both performed based on blocks, offering high security and efficiency[44].

The AES algorithm's encryption and decryption operations involve the following four steps:

1) SubBytes:uses a substitution table called the S-Box to replace each byte in the input block with its corresponding S-Box value.
2) ShiftRows: cyclically shifts each row to the left, where the number of shifts depends on the block size and the row number.
3) MixColumns:maps each input column vector in the block to an output column vector, enhancing data irregularity and interactivity.
4) AddRoundKey:performs bitwise XOR between the round key used in the encryption process and the current state matrix.

AES algorithm offers advantages in both security and efficiency and finds extensive applications in various fields such as network communication, data storage, financial transactions, etc. Its key benefits include high security, efficiency, and reliability.

3.4. FASR

$FASR$ is a novel Feature-Aware Stateful Routing method designed to reduce system overhead in distributed environments and maintain a high deduplication rate[45]. The $FASR$ system primarily consists of three working nodes, $Client$, $Master$, and $Data\ nodes$. The $Client$ is responsible for receiving data streams and sending requests for deduplication and recovery. The $Master\ node$ maintains the mapping between data blocks and file fingerprints to provide the metadata required for deduplication and recovery. While the $Data\ nodes$ primarily handle data retrieval and storage.

4. System Model Design

This section mainly introduces the architecture of the medical record information management system based on blockchain and access control. Section 4.1 presents the fundamental framework of the system, while Section 4.2 provides a detailed description of the system design process. The proposed solution's system flow is illustrated in Section 4.3.

4.1. Basic Architecture

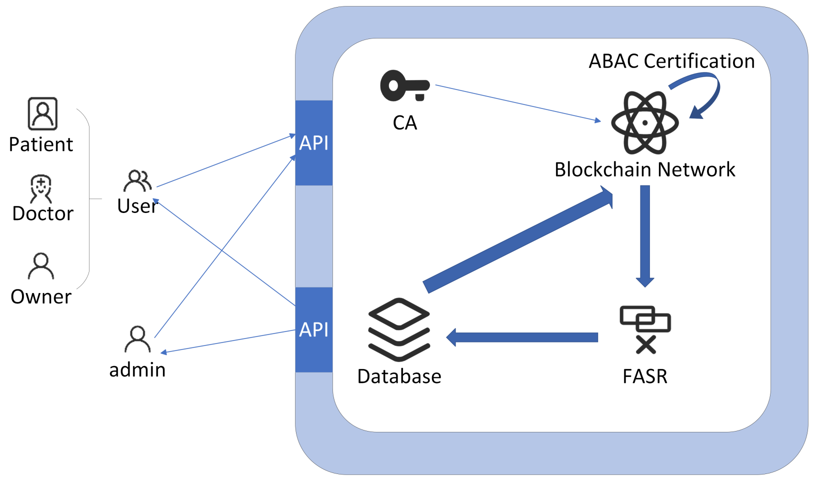


Fig. 2. System Architecture Diagram

The system architecture consists of three essential components, users, the blockchain network, and $FASR\ nodes$. After completing identity authentication, each user is issued a digital certificate by the Certificate Authority (CA). Only users with digital certificates have access privileges to the blockchain system, which assures the fundamental security of the blockchain. After receiving a user's access information, the blockchain utilizes smart contracts to determine the legitimacy of the access information and checks if the user has the necessary permissions to execute the operation. And then it transmits a corresponding feedback to the user. The detailed architecture of the system is illustrated in Figure2.

The system users mainly consist of backend institutions, administrators, doctors, and patients. Different users have different access privileges. Backend institutions undertakes to package and upload medical record information. Additionally, they play a supervisory role to prevent malicious attacks or illegal operations by users. Administrators are only in charge of creating doctor and patient accounts, and the access permissions for doctor and patient users are system-defined and independent of the administrators. Doctors have permissions to view medical records, add patient information and medical record details, and upload modifications to medical records. As for patients, they can only access their own medical records. Each role has its own responsibilities and cannot exceed their authorized boundaries.

The blockchain network is built on Hyperledger Fabric and primarily implements the following functionalities.

1) The CA issues digital certificates to participating nodes in the blockchain network for identity authentication and communication encryption.
2) Smart contracts automatically upload data to the database after each operation and supervise user access requests based on access control rules.

In this paper, $FASR$ is used to compress medical record data in a lean format and store them in the blockchain database, so as to reduce redundant operations on duplicate data during interactions with the blockchain system.

4.2. Access Control Model Design

In the supply chain of medical record information, a finer-grained distinction of role permissions and dynamic management of different access rights are required. Therefore, in this paper, we adopt the $ABAC$ model as the overall system architecture and define User Attribute (UA), Medical Record Attribute (MA), Permission Attribute (PA), and Environment Attribute (EA) separately. Considering the potential overlapping operations among different users in UA, we introduce the $RBAC$ policy to bundle all relevant operations into Roles and take Role as an attribute in the $ABAC$ model. This design not only achieves dynamic fine-grained access to complex permissions in the traceability supply chain but also resolves the problem of redundant policy storage by utilizing the $RBAC$ model.

RBAC Design

This subsection primarily introduces all user entities in the system. Three identity roles, Administrator, Doctor, and Patient, are designed. Table2 displays the access control permissions for different users.

1) The Administrator is the manager of the entire system, and every user who wants to access the system needs to be granted different user roles by the Administrator. Therefore, the Administrator can assign new users either the Doctor role or the Patient role and provide each user with a unique identification ID.

Table 2. Access Control Permissions for Different Users

	FindByDid	CreDoc	CrePat	AddUit	AddAit	AddTit
Owner		✓	✓			
Doctor	✓			✓	✓	✓
Patient	✓					

2) The Doctor plays a crucial and pivotal role in the entire system, as only doctors have the authority to modify and add patient information. Doctors diagnose patients' conditions, determine their medical record information, and promptly upload the medical records to the system.
3) The Patient is a vital component of the entire system. Patients can access the system and inquire about their medical record information using their own ID information. To ensure system security, doctors can only operate on medical records with the patient's permission, using the medical record identification number.

ABAC Design

Based on the actual application scenario of the medical traceability system, we define each attribute UA, MA, EA, and PA in the ABAC model as follows.

UA denotes the attributes of user entities and its specific attributes are shown in Table3. $UserId$ serves as the unique identifier for each user, and a user has only one $UserId$. $Role$ represents the user's role, which can be Administrator, Doctor, or Patient. Different user roles correspond to different system access permissions. The definition of roles has been elaborated in the $RBAC$ section.

Table 3. UA Attribute Definition Table

Name	Definition
UserId	Unique number for each user
Role	Identification of user identity
Group	The group to which the user belongs

MA represents the entity attributes of medical record information and its detailed attributes are shown in Table4. Did serves as the unique identifier for each medical record and a medical record has only one Did. $Disease$ represents the medical record form in which specific information about the patient's medical history, medication, etc., is recorded.

EA is the environmental information. In our paper, EA's attribute denotes $AllowedTime$, a reasonable amount of time for temporary users to access. If the requested time by a temporary user exceeds the validity period of the permission, the user's operation will be denied.

Table 4. MA Attribute Definition Table

Name	Definition
Did	Unique number for each medical record
OwnerId	Patients with medical records
Name	The patient's name
Age	The patient's age
Disease	medical records
HisUpdate	A list of medical record modifications
HisAdd	A list of medical records is added

PA represents the system's access control policy. If it is true, access is allowed; otherwise, access is denied. Eq1precisely describes the access permission.

$$PA = \begin{cases} true & allow \\ false & deny \end{cases} \tag{1}$$

$ASAO$ shows the mapping relationship between users and medical records. To achieve fine-grained permission allocation, we utilize $ASAO$ as the index for storing access control policies in the blockchain. $ASAO$ is comprised of $AS.UserID$, $AS.Role$, and $AO.Did$, and establishes a one-to-one correspondence between different user identities and medical records. Furthermore, the AES algorithm is employed to encrypt $ASAO$ and the encrypted index is stored in the blockchain as the index for access control policies, so as to achieve the uniqueness of the index. $ASAO$ is calculated based on eq2.

$$index \xleftarrow{AES} ASAO \leftarrow \{UA.UserId, MA.Did, MA.UserRole\} \tag{2}$$

The access control policy, Policy, consists of four main components: UA, MA, PA, and EA.

$$Policy = \{UA, MA, PA, EA\} \tag{3}$$

To enhance the efficiency of access control queries and validations, Policy is stored in the blockchain ledger in the form of $\langle Index, Policy \rangle$.

FASR Design

$FASR$ technology is employed to perform deduplication on data that needs to be stored in the blockchain, reducing the storage burden on the blockchain, minimizing data redundancy, and improving storage efficiency. $FASR$ mainly comprises three working nodes, namely the $Client$, the $Master\ node$, and the $Data\ nodes$. The primary workflow in this paper includes the following steps.

1) The $Client$ transfers data to the $Master\ node$.

$$Client \xrightarrow{data} Master \longrightarrow node\ feature\ list \tag{4}$$

2) The $Master\ node$ filters out nodes that do not contain the same type of data, based on the node feature table, and selects the node with the highest data type storage capacity.

$$Master \xrightarrow{node\ feature\ list} Nodes \tag{5}$$

3) $FASR$ calculates the similarity between the data and the selected node. Based on the similarity and node storage capacity, the target node is chosen.

$$Nodes \overset{similarity}{\longrightarrow} Target\ Node \tag{6}$$

4) Using data similarity and locality, local duplicate data removal is performed, and the processed data is stored in the blockchain.

$$Target\ Node \overset{deduplicate}{\longrightarrow} Data \longrightarrow Blockchain \tag{7}$$

Smart Contract Design

In addition to implementing access control, smart contracts are also related to the storage of medical record information. To achieve a medical record traceability system based on access control, three types of smart contracts, Access Contract (AC), Device Contract (DC), and Policy Contract (PC), are formulated.

1) AC: It implements access control for user requests within the blockchain network. AC verifies user permissions based on the requests submitted by users and their roles. AC serves as an interface for user access. Upon receiving users' request, it performs the necessary authentication and then calls other relevant contracts for specific operations. The AC mainly includes the following methods.
$CheckAccess()$: This method is the core function of AC, responsible for validating the access information sent by users and executing corresponding operations based on specific requests. $CheckAccess()$ can be divided into three steps, confirming input parameters, authenticating access control policies, and executing operation instructions. Firstly, users input the access parameters to submit the operation request to the blockchain. As shown in eq8, $AccessAvg$ is composed of $ASAO$,$Operation$ and $OperationContent$. and are name and content of the operation, respectively.

$$AccessAvg \longleftarrow (ASAO, Operation, OperationContent) \tag{8}$$

Among them, there are six types of operation, namely $FindByDid$, $CrePat$, $CreDoc$, $AddTit$, $AddUit$, and $AddAit$, representing the operations to find medical record information, create a patient user, create a doctor user, add medical record details, add medical record modification information, and add medical record additional information, respectively.

$$AccessAvg \longleftarrow (ASAO, FindByDid) \tag{9}$$

$$AccessAvg \longleftarrow (ASAO, Credoc, UseId) \tag{10}$$

$$AccessAvg \longleftarrow (ASAO, AddUit, Iid, Result) \tag{11}$$

$$AccessAvg \longleftarrow (ASAO, AddTit, Iid, Disease, Medicine) \tag{12}$$

Algorithm 1 Part of CheckAccess(): The Specific Process of Handling User Operations

```
Require: $Args$
Ensure: $Response\ or\ Error$
if $Operations == FindByDid$ then
    if $len(Args)! = 2$ then
        return $Error : Args\ Invalid$
    end if
    if AS.Role $\notin \langle Docter, Patient\rangle$ then
        return $Error : Identity\ Invalid$
    end if
    $MaDid \leftarrow InvokeChaincode(DC, FindByDid, AO.Did)$
    $\langle Resp, Error\rangle \leftarrow UnMamarshal(MaDid)$
    if $Did == NullorError! = Null$ then
        return $Error : NofindDid$
    end if
    return $Did.HisAdd$
end if
if $Operations == CreDoc$ then
    if $len(Args)! = 3$ then
        return $Error : Args\ Invalid$
    end if
    if $AS.Role! = Owner$ then
        return $Error : Identity\ Invalid$
    end if
    $NewABACPolicy \leftarrow \langle NewAS, AO, AP, AE\rangle$
    $Reponse \leftarrow InvokeChaincode(PC, AddPolicy, NewABACPolicy)$
    return Reponse
end if
if Operations == AddTit then
    if len(Args)!=5 then
        return Error:Args Invalid
    end if
    if $AS.Role \notin \langle Docter\rangle$ then
        return Error: Identity Invalid
    end if
    $Reponse \leftarrow InvokeChaincode(DC, AddMedicalRe, AO.Did, ArgsAdd)$
    return $Reponse$
end if
```

As showen in Eq 9,10,11,12, the number of parameters in $AccessAvg$ during the user request process can only be 2, 3, 4, or 5. Therefore, before validating the access request policy, it is essential to check the number of parameters in $AccessAvg$ and reject access with invalid parameters, which significantly improves the execution speed of AC policies.

Next, the value of AP in the AC policy needs to be determined. If it is 1, access is permitted; otherwise, access is denied.

$$Policy.PA = \begin{cases} 1 & allow \\ 0 & deny \end{cases} \tag{13}$$

In the blockchain, different operation permissions are assigned to different roles. After the user are verified, the system further validates the remaining $ABAC$ attributes based on the user's identity and the operation for request. Upon successful verification, the relevant smart contract is automatically invoked to execute the corresponding operations. Different operations require different $ABAC$ attribute to validate and different methods to invoke. The detailed smart contract design is shown in Algorithm 1.

2) DC: It primarily handles the management of medical record information on the blockchain. Its main functionalities include uploading medical record information and adding medical record entries. DC does not directly open operations to patient or doctor users. AC verifies users' queries or modification requests to devices based on the access control rules and then automatically calls DC to update or return the modified data. According to the main functions of the data center, the following methods are designed for it:

   $FindByDid()$: The main function of this method is to query medical record information through the unique index, Did, and return detailed data of the medical record.

   $AddMedRecord()$: Adding medical record information, including the patient's condition, medication, etc.

   $AddUitem()$: Adding modifications to the medical record, including the reason for the modification and the modified record entries.

3) PC: It takes charge of managing the entire system's access control policies. When users employ AC to initiate operations on devices, the system automatically calls PC to query and return the user's access control policies for AC authentication. Besides, when users perform AC-related functions, PC is invoked to update the access control policies if any changes occur in the access control policies. Similar to DC, PC is not directly accessible to users, but only provides interfaces for communication between users.

Algorithm 2 Part of AddU():Add Policy

```
Require: $Args$
Ensure: $True\ or\ Error$
1: $\langle AS, AO, AE, AP\rangle \leftarrow$ Args
2: $Policyld \leftarrow AES(AS.Userd + AO.Did + AS.Role)$
3: $ExistPolicy \leftarrow InvokeChaincode(Pc, QueryPolicy, Policyld)$
4: if $ExistPolicy! = False$ then
5:     return $Error : Policy\ HaveAlreadyExisted$
6: end if
7: $\langle MaPolicy, err\rangle \leftarrow Marshal(AS, AO, AP, AE)$
8: if $err! = True$ then
9:     return $Error$
10: end if
11: $resp \leftarrow APIstub.PutState(Policyld, MaPolicy)$
12: if $err! = True$ then
13:     return $Error : Failed\ To\ Add\ Policy$
14: end if
15: return $true$
```

$AddPolicy()$: It aims to upload permission policy information. The policy is stored on the blockchain with the index being the hashed value of encrypted ASAO. When uploading a policy, the blockchain is first queried for the existence of the policy index. If it already exists, the policy cannot be added. Algorithm 2outlines the detailed process of $AddPolicy()$.
$QueryPolicy()$: Before performing any operation in the system, users must undergo permission verification. This function returns the user's access policy on the blockchain for AC to verify the user's access permissions. Based on the input policy index, it searches the blockchain and returns the corresponding search result.
$UpdatePolicy()$: This method does not support nor provide interfaces for external client invocation. When some of the users' operations involve changes to their access control policies, the smart contract automatically calls this method to update and maintain the access control policy ledger on the blockchain.

4.3. Execution Steps

As illustrated in Figure3, the specific workflow of this proposed solution mainly consists of four parts. The symbols used are shown in Table5.

Table 5. Used Symbols

Notations	Definition
CA	Certificate Authority
Cert	Certificate file
Key	Key in Hyperledger Fabric
Ledger	Ledger in Hyperledger Fabric
CC	Chaincode in Hyperledger Fabric
Channel	Passage in Hyperledger Fabric
f_x	Functions defined in source code
Ledger	Ledger in Hyperledger Fabric
SDB	State Database in Hyperledger Fabric
ABAC	Attribute based Access Control
ABAC	Feature-aware de-duplication
Cli	Blockchain system client

Part 1: This process involves building the blockchain network and deploying the chaincode. These fundamental steps are carried out by the backend institution. Part 1 is further divided into the following steps.
Step 1: CA tools are used to generate certificates and keys for identity authentication and encryption within the Fabric network.

$$CA = \{Cert, Key\} \tag{14}$$

Step 2: Using configtxgen and $PeerChannel$ tools to create channels and join peer nodes.

$$\{blockchain, ledger\} \xrightarrow{jion} Channel \tag{15}$$

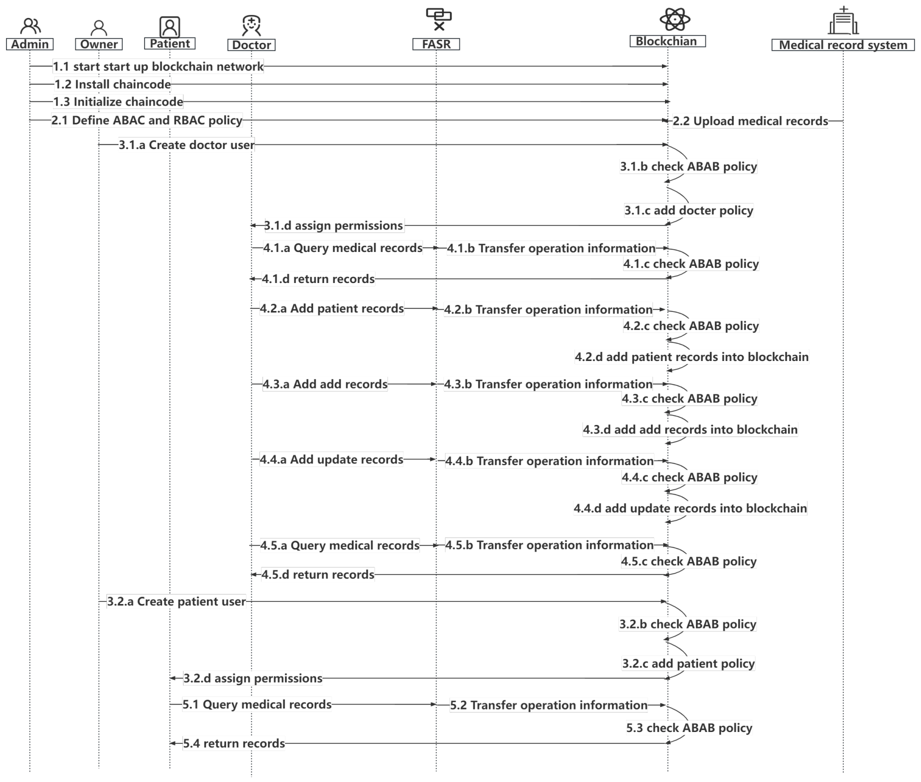

Admin
Owner
Patient
Doctor
FASR
Blockchian
Medical record system
1.1 start start up blockchain network
1.2 Install chaincode
1.3 Initialize chaincode
2.1 Define ABAC and RBAC policy
2.2 Upload medical records
3.1.a Create doctor user
3.1.b check ABAB policy
3.1.c add docter policy
3.1.d assign permissions
4.1.a Query medical records
4.1.b Transfer operation information
4.1.c check ABAB policy
4.1.d return records
4.2.a Add patient records
4.2.b Transfer operation information
4.2.c check ABAB policy
4.2.d add patient records into blockchain
4.3.a Add add records
4.3.b Transfer operation information
4.3.c check ABAB policy
4.3.d add add records into blockchain
4.4.a Add update records
4.4.b Transfer operation information
4.4.c check ABAB policy
4.4.d add update records into blockchain
4.5.a Query medical records
4.5.b Transfer operation information
4.5.c check ABAB policy
4.5.d return records
3.2.a Create patient user
3.2.b check ABAB policy
3.2.c add patient policy
3.2.d assign permissions
5.1 Query medical records
5.2 Transfer operation information
5.3 check ABAB policy
5.4 return records

Fig. 3. System Architecture Diagram

Step 3: Deploy and execute smart contract using $Chaincode$ tool

$$ChainCode(f_x) \longrightarrow CC \tag{16}$$

Part 2: After creating the policies, the administrator needs to save them to the blockchain.
Step 1: The administrator and users set access control policies based on UA, MA, PA, and EA.

$$Set\,(UA, MA, PA, EA) \longrightarrow ABAC \tag{17}$$

Step 2: The administrator uploads the developed access control policies to the blockchain network.

$$deploy\,(ABAC) \longrightarrow Constract \tag{18}$$

Step 3:The administrator runs PC to execute operations such as adding and modifying policies, and saves the final policy values to SDB and the $ledge$r.

$$PC\,(ABAC) \longrightarrow \{SDB, Ledger\} \tag{19}$$

Part 3:In this part, users' access requests are first uploaded to $FASR$ to reduce data redundancy before storing the processed data in the blockchain.
Step 1: Users upload medical record information to $FASR$.

$$upload\,(MedicalRecords) \longrightarrow FASR \tag{20}$$

Step 2:$FASR$ deduplicates the medical record information and stores it in the blockchain.

$$FASR\,(MedicalRecords) \longrightarrow blockchain \tag{21}$$

Part 4: This part is the process of accessing medical information based on attribute-based access control and can be divided into four specific steps.
Step 1: The user initiates a request to access medical data.

$$Request \longrightarrow blockchain \tag{22}$$

Step 2: Upon receiving the user's request, AC policies are invoked to verify if the user can access the data.

$$AC\,(request) \longrightarrow \begin{cases} 1 & allow \\ 0 & deny \end{cases} \tag{23}$$

Step 3: The user's request is stored after deduplication through $FASR$.

$$blockchain\,(Request) \longrightarrow FASR \tag{24}$$

Step 4: The result of the user's request is returned.

$$Response\,(MedicialRecord) \longrightarrow Cli \tag{25}$$

5. Experimental Results and Analysis

This section mainly describes the entire experimental process. In Section5.1, the results of the simulation experiment are presented, and in Section5.2, we analyze the experimental data. To highlight the innovation of this solution, we conducted a comparison with relevant literature in Section5.3. The simulation experiment environment for this solution is presented in Table6.

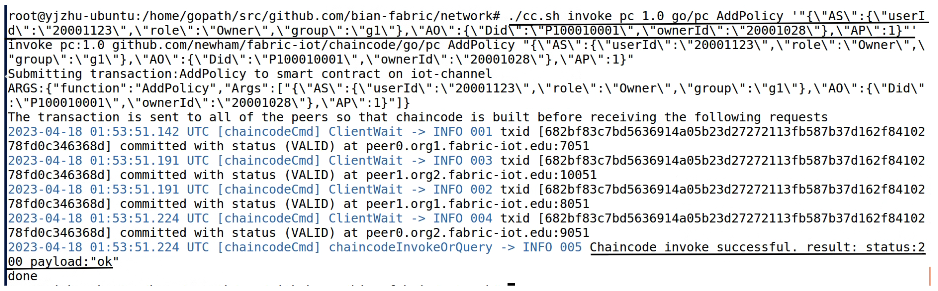
Table 6. Experimental Environment

Experimental environment	
CPU	Intel core i5-7300
Memory	8 GB
Docker	v19.03.2
Docker-compose	v1.24.1
Node	v12.12.0
Hyperledger Fabric	v1.4.3
Golang	v1.13.5

5.1. Experimental Results

To validate the data security, decentralization of permission management, dynamic nature of permission updates, and the immutability of permission transfer records in the proposed system, three users are designed for the experiment, namely Owner, Doctor, and Patient, with IDs ”20001123”, ”19991123”, and ”20001028”, respectively. A medical record information P1 with a unique index Did of ”P100010001” is used. Firstly, the medical institution adds the policies to the blockchain network and the experimental results for policy upload are shown in Figure4. The main purpose of uploading policies is to configure the administrator’s basic information on the blockchain network.

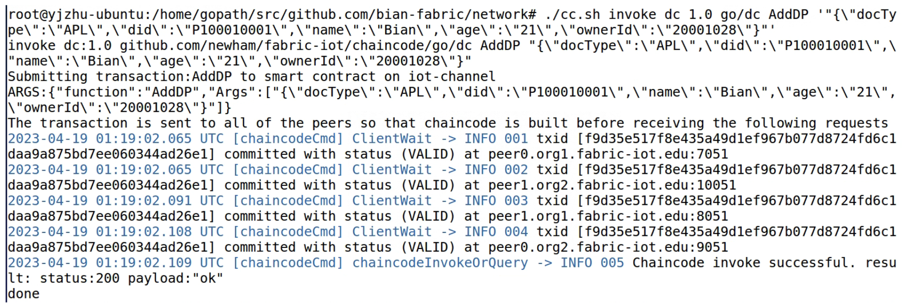
In addition to uploading administrator information, the medical institution’s backend also needs to package and upload P1 to the blockchain system. In this experiment, a set of medical record information is simulated. Figure5 shows the process of of requesting the upload of medical record information and returning parameters.



```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke pc 1.0 go/pc AddPolicy '"{\"AS\":{\"userI
d\":\"20001123\",\"role\":\"Owner\",\"group\":\"g1\"},\"AO\":{\"Did\":\"P100010001\",\"ownerId\":\"20001028\"},\"AP\":1}"'
invoke pc:1.0 github.com/newham/fabric-iot/chaincode/go/pc AddPolicy "{\"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\",\
"group\":\"g1\"},\"AO\":{\"Did\":\"P100010001\",\"ownerId\":\"20001028\"},\"AP\":1}"
Submitting transaction:AddPolicy to smart contract on iot-channel
ARGS:{"function":"AddPolicy","Args":["{\"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\",\"group\":\"g1\"},\"AO\":{\"Did\"
:\"P100010001\",\"ownerId\":\"20001028\"},\"AP\":1}"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-18 01:53:51.142 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [682bf83c7bd5636914a05b23d27272113fb587b37d162f84102
78fd0c346368d] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-18 01:53:51.191 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [682bf83c7bd5636914a05b23d27272113fb587b37d162f84102
78fd0c346368d] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-18 01:53:51.191 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [682bf83c7bd5636914a05b23d27272113fb587b37d162f84102
78fd0c346368d] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-18 01:53:51.224 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [682bf83c7bd5636914a05b23d27272113fb587b37d162f84102
78fd0c346368d] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-18 01:53:51.224 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. result: status:2
00 payload:"ok"
done
```

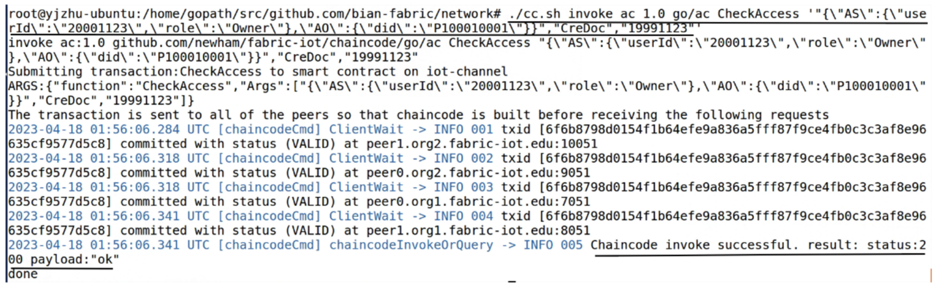
Fig. 4. Upload Policy

In the system design presented in Section 4, we have provided a detailed overview of the access control permissions for each role. The medical institution is only required to upload blockchain policies and package medical record information. Subsequent simulated operations are carried out by the administrator, patient, and doctor. The Owner creates Doctor and Patient users by calling $CheckAccess()$ from AC, the system automatically verifies the access control permissions using $QueryPolicy()$ from PC. After successful verification, the $AddUser()$ is invoked, as shown in Figure6. In Figure7, the Owner creates the Patient role.



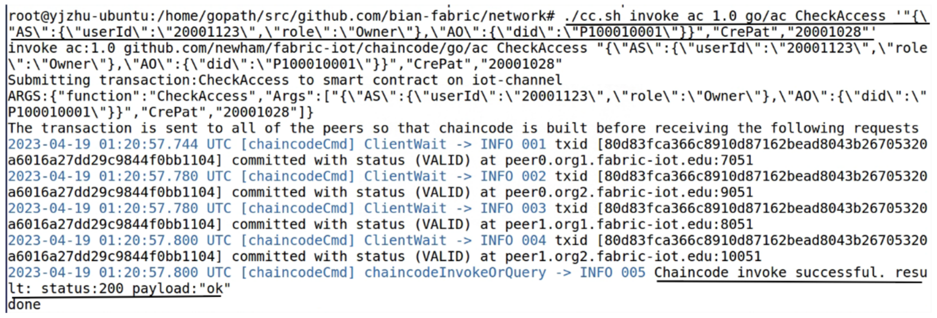
```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke dc 1.0 go/dc AddDP '"{\"docTy
pe\":\"APL\",\"did\":\"P100010001\",\"name\":\"Bian\",\"age\":\"21\",\"ownerId\":\"20001028\"}"'
invoke dc:1.0 github.com/newham/fabric-iot/chaincode/go/dc AddDP "{\"docType\":\"APL\",\"did\":\"P100010001\",\
"name\":\"Bian\",\"age\":\"21\",\"ownerId\":\"20001028\"}"
Submitting transaction:AddDP to smart contract on iot-channel
ARGS:{"function":"AddDP","Args":["{\"docType\":\"APL\",\"did\":\"P100010001\",\"name\":\"Bian\",\"age\":\"21\",
\"ownerId\":\"20001028\"}"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-19 01:19:02.065 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [f9d35e517f8e435a49d1ef967b077d8724fd6c1
daa9a875bd7ee060344ad26e1] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-19 01:19:02.065 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [f9d35e517f8e435a49d1ef967b077d8724fd6c1
daa9a875bd7ee060344ad26e1] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-19 01:19:02.091 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [f9d35e517f8e435a49d1ef967b077d8724fd6c1
daa9a875bd7ee060344ad26e1] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-19 01:19:02.108 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [f9d35e517f8e435a49d1ef967b077d8724fd6c1
daa9a875bd7ee060344ad26e1] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-19 01:19:02.109 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. resu
lt: status:200 payload:"ok"
done
```

Fig. 5. Upload Medical Record Information



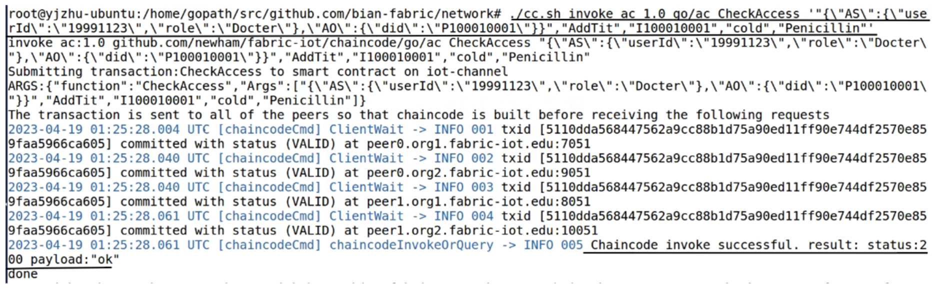
```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke ac 1.0 go/ac CheckAccess '"{\"AS\":{\"use
rId\":\"20001123\",\"role\":\"Owner\"},\"AO\":{\"did\":\"P100010001\"}}","CreDoc","19991123"'
invoke ac:1.0 github.com/newham/fabric-iot/chaincode/go/ac CheckAccess "{\"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\"
},\"AO\":{\"did\":\"P100010001\"}}","CreDoc","19991123"
Submitting transaction:CheckAccess to smart contract on iot-channel
ARGS:{"function":"CheckAccess","Args":["{\"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\"},\"AO\":{\"did\":\"P100010001\"
}}","CreDoc","19991123"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-18 01:56:06.284 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [6f6b8798d0154f1b64efe9a836a5fff87f9ce4fb0c3c3af8e96
635cf9577d5c8] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-18 01:56:06.318 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [6f6b8798d0154f1b64efe9a836a5fff87f9ce4fb0c3c3af8e96
635cf9577d5c8] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-18 01:56:06.318 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [6f6b8798d0154f1b64efe9a836a5fff87f9ce4fb0c3c3af8e96
635cf9577d5c8] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-18 01:56:06.341 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [6f6b8798d0154f1b64efe9a836a5fff87f9ce4fb0c3c3af8e96
635cf9577d5c8] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-18 01:56:06.341 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. result: status:2
00 payload:"ok"
done
```

Fig. 6. Create Doctor User



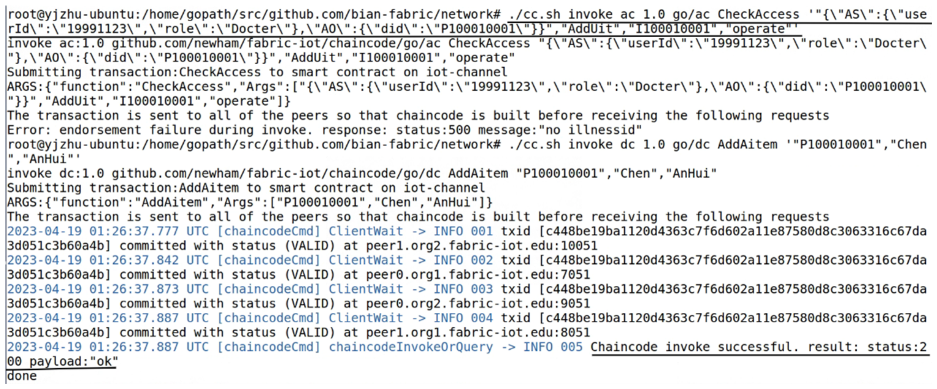
```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke ac 1.0 go/ac CheckAccess '"{\
"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\"},\"AO\":{\"did\":\"P100010001\"}}","CrePat","20001028"'
invoke ac:1.0 github.com/newham/fabric-iot/chaincode/go/ac CheckAccess "{\"AS\":{\"userId\":\"20001123\",\"role
\":\"Owner\"},\"AO\":{\"did\":\"P100010001\"}}","CrePat","20001028"
Submitting transaction:CheckAccess to smart contract on iot-channel
ARGS:{"function":"CheckAccess","Args":["{\"AS\":{\"userId\":\"20001123\",\"role\":\"Owner\"},\"AO\":{\"did\":\"
P100010001\"}}","CrePat","20001028"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-19 01:20:57.744 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [80d83fca366c8910d87162bead8043b26705320
a6016a27dd29c9844f0bb1104] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-19 01:20:57.780 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [80d83fca366c8910d87162bead8043b26705320
a6016a27dd29c9844f0bb1104] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-19 01:20:57.780 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [80d83fca366c8910d87162bead8043b26705320
a6016a27dd29c9844f0bb1104] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-19 01:20:57.800 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [80d83fca366c8910d87162bead8043b26705320
a6016a27dd29c9844f0bb1104] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-19 01:20:57.800 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. resu
lt: status:200 payload:"ok"
done
```

Fig. 7. Create Patient User



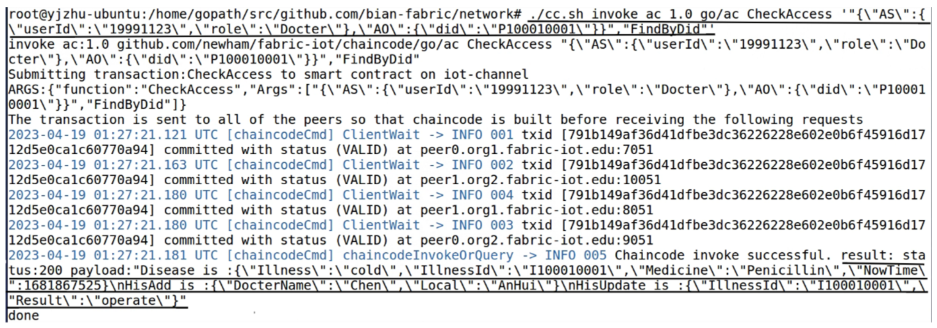
```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke ac 1.0 go/ac CheckAccess '"{\"AS\":{\"use
rId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P100010001\"}}","AddTit","I100010001","cold","Penicillin"'
invoke ac:1.0 github.com/newham/fabric-iot/chaincode/go/ac CheckAccess "{\"AS\":{\"userId\":\"19991123\",\"role\":\"Docter\
"},\"AO\":{\"did\":\"P100010001\"}}","AddTit","I100010001","cold","Penicillin"
Submitting transaction:CheckAccess to smart contract on iot-channel
ARGS:{"function":"CheckAccess","Args":["{\"AS\":{\"userId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P100010001\
"}}","AddTit","I100010001","cold","Penicillin"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-19 01:25:28.004 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [5110dda568447562a9cc88b1d75a90ed11ff90e744df2570e85
9faa5966ca605] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-19 01:25:28.040 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [5110dda568447562a9cc88b1d75a90ed11ff90e744df2570e85
9faa5966ca605] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-19 01:25:28.040 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [5110dda568447562a9cc88b1d75a90ed11ff90e744df2570e85
9faa5966ca605] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-19 01:25:28.061 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [5110dda568447562a9cc88b1d75a90ed11ff90e744df2570e85
9faa5966ca605] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-19 01:25:28.061 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. result: status:2
00 payload:"ok"
done
```

Fig. 8. Add Medical Record Information



```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke ac 1.0 go/ac CheckAccess '"{\"AS\":{\"use
rId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P100010001\"}}","AddUit","I100010001","operate"'
invoke ac:1.0 github.com/newham/fabric-iot/chaincode/go/ac CheckAccess "{\"AS\":{\"userId\":\"19991123\",\"role\":\"Docter\
"},\"AO\":{\"did\":\"P100010001\"}}","AddUit","I100010001","operate"
Submitting transaction:CheckAccess to smart contract on iot-channel
ARGS:{"function":"CheckAccess","Args":["{\"AS\":{\"userId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P100010001\
"}}","AddUit","I100010001","operate"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
Error: endorsement failure during invoke. response: status:500 message:"no illnessid"
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke dc 1.0 go/dc AddAitem '"P100010001","Chen
","AnHui"'
invoke dc:1.0 github.com/newham/fabric-iot/chaincode/go/dc AddAitem "P100010001","Chen","AnHui"
Submitting transaction:AddAitem to smart contract on iot-channel
ARGS:{"function":"AddAitem","Args":["P100010001","Chen","AnHui"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-19 01:26:37.777 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [c448be19ba1120d4363c7f6d602a11e87580d8c3063316c67da
3d051c3b60a4b] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-19 01:26:37.842 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [c448be19ba1120d4363c7f6d602a11e87580d8c3063316c67da
3d051c3b60a4b] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-19 01:26:37.873 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [c448be19ba1120d4363c7f6d602a11e87580d8c3063316c67da
3d051c3b60a4b] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-19 01:26:37.887 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [c448be19ba1120d4363c7f6d602a11e87580d8c3063316c67da
3d051c3b60a4b] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-19 01:26:37.887 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. result: status:2
00 payload:"ok"
done
```

Fig. 9. Add Medical Record Modification Information



```
root@yjzhu-ubuntu:/home/gopath/src/github.com/bian-fabric/network# ./cc.sh invoke ac 1.0 go/ac CheckAccess '"{\"AS\":{
\"userId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P100010001\"}}","FindByDid"'
invoke ac:1.0 github.com/newham/fabric-iot/chaincode/go/ac CheckAccess "{\"AS\":{\"userId\":\"19991123\",\"role\":\"Do
cter\"},\"AO\":{\"did\":\"P100010001\"}}","FindByDid"
Submitting transaction:CheckAccess to smart contract on iot-channel
ARGS:{"function":"CheckAccess","Args":["{\"AS\":{\"userId\":\"19991123\",\"role\":\"Docter\"},\"AO\":{\"did\":\"P10001
0001\"}}","FindByDid"]}
The transaction is sent to all of the peers so that chaincode is built before receiving the following requests
2023-04-19 01:27:21.121 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [791b149af36d41dfbe3dc36226228e602e0b6f45916d17
12d5e0ca1c60770a94] committed with status (VALID) at peer0.org1.fabric-iot.edu:7051
2023-04-19 01:27:21.163 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [791b149af36d41dfbe3dc36226228e602e0b6f45916d17
12d5e0ca1c60770a94] committed with status (VALID) at peer1.org2.fabric-iot.edu:10051
2023-04-19 01:27:21.180 UTC [chaincodeCmd] ClientWait -> INFO 004 txid [791b149af36d41dfbe3dc36226228e602e0b6f45916d17
12d5e0ca1c60770a94] committed with status (VALID) at peer1.org1.fabric-iot.edu:8051
2023-04-19 01:27:21.180 UTC [chaincodeCmd] ClientWait -> INFO 003 txid [791b149af36d41dfbe3dc36226228e602e0b6f45916d17
12d5e0ca1c60770a94] committed with status (VALID) at peer0.org2.fabric-iot.edu:9051
2023-04-19 01:27:21.181 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 005 Chaincode invoke successful. result: sta
tus:200 payload:"Disease is :{\"Illness\":\"cold\",\"IllnessId\":\"I100010001\",\"Medicine\":\"Penicillin\",\"NowTime\
":1681867525}\nHisAdd is :{\"DocterName\":\"Chen\",\"Local\":\"AnHui\"}\nHisUpdate is :{\"IllnessId\":\"I100010001\",\
"Result\":\"operate\"}"
done
```

Fig. 10. Query Medical Record Information

The Doctor adds medical record information P1. Here, the Doctor user uses Check-Access(). However, when the system uses $CheckAccess()$, it also automatically calls $QueryPolicy()$ from PC and $AddMedRecords()$ from DC to perform access control authorization verification and medical record information addition, respectively. As shown in Figure8, the Doctor user successfully adds the medical record information.

For adding modification and additional information to the medical record, as shown in Figure9, the Doctor user adds modification information. The process of adding additional information is similar and also utilizes $CheckAccess()$. Similarly, the blockchain system will call the $QueryPolicy()$ method from PC and the $FindByDid()$ and $AddUitem()$ methods from DC. After adding the information, the Doctor can query the complete medical record information, as shown in Figure10.

5.2. Performance Testing

To validate the actual performance of the proposed system, three sets of simulated experiments are conducted. In the first and second sets of experiments, the number of virtual concurrent clients is set to 100, 200, 400, 600, 800, and 1000, and the time consumption and throughput of PC and AC are measured, as shown in Figs.11 and 12, respectively. In the second set of experiments, the number of virtual concurrent clients is set to 100, 200, 300, 400, 500, 600, 800, and 1000, and the time consumption and throughput of DC are measured, as shown in Fig.13.

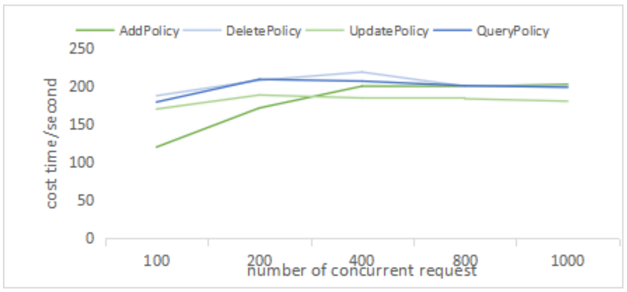


Fig. 11. Throughput of PC under different concurrent requests.

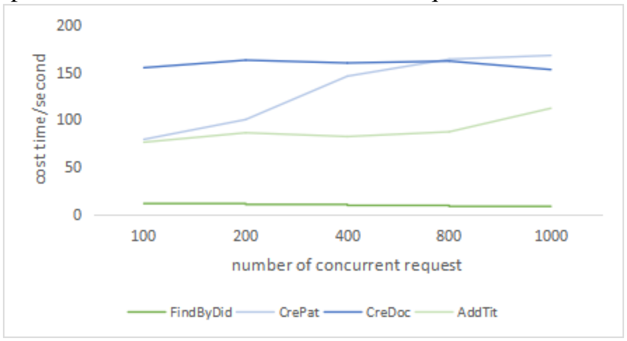


Fig. 12. Throughput of AC under different concurrent requests.

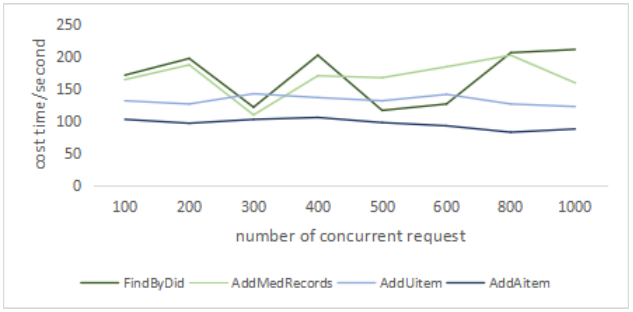


Fig. 13. Throughput of DC under different concurrent requests.

Based on the above experimental results, the following conclusions can be drawn. The throughput of adding and updating operations is lower than that of querying and deleting operations. When the number of concurrent requests reaches a certain value, the throughput does not significantly decrease. This is because when the number of connections in the blockchain network's connection pool reaches its limit, the system's throughput tends to stabilize.

We compare the throughput of the AC method by setting different numbers of nodes, ranging from 100 to 1000. We observe that the throughput of $FindByDid()$is significantly lower than the other three methods.

The throughput of operations in AC is generally lower than that of operations in DC and PC. In fact, operations in AC often require calling operations in DC and PC, making the complexity of operations in AC higher than in DC and PC.

5.3. Comparison with Existing Works

To highlight the innovation of our proposed system, we compare it with other relevant systems, as shown in Table7. In terms of data security, protecting privacy information requires enhanced data security measures. But Literature[46] overlooks data security issues. Literatures[47]and[48] utilize consortium blockchain to store private data, which indeed improves data security to some extent. However, solely relying on blockchain features is not sufficient to ensure data security. Storing data in the blockchain does not necessarily achieve dynamic and fine-grained access to medical records. Compared to the method in[47], the work in[48] proposes a blockchain-based medical information sharing platform, enabling medical information to be stored, shared, and trusted across distributed networks. Regarding decentralized access control, accessing medical records is often closely related to user roles in practical applications. Literatures[46],[47],and [48]merely store medical records in the blockchain without distinguishing fine-grained access to medical information. To overcome the limitations mentioned above, we combine the strengths of ABAC and RBAC models and utilize the features of AES algorithm to achieve fine-grained and dynamic permission management,and use FASR deduplication technology to reduce data redundancy. And the access control model is implemented on the basis of smart contracts, ensuring real-time monitoring of on-chain data and enhancing the overall system's security.

Table 7. Comparison with Existing Solutions

Paper	[46]	[47]	[48]	Ours
Access control	ABAC	No	ABAC	ABACandRBAC
Security	No	No	No	AES
Smart contract	No	Yes	Yes	Yes
Blockchain platform	No	Hyperledger Fabric	Hyperledger Fabric	Hyperledger Fabric
Storage strategy	No	No	No	FASR

6. Conclusions

This paper combines blockchain technology with ABAC and RBAC models to effectively break the information silos in medical data, and ensure the security of medical records while enabling data sharing. The FASR technology is used to handle medical records, reducing data redundancy and alleviating blockchain storage pressure. Moreover, the proposed solution utilizes a distributed architecture to achieve dynamic and fine-grained role division, integrating blockchain with real-world applications for efficient medical record management. The deployment and invocation process of chaincode are described in detail, and the feasibility of the system is demonstrated through experiments.

This model is expected to be applied to practical medical scenarios, and with the rise of the concept of meta-universe, traditional medical treatment should also keep pace with The Times. Led by relevant authorities, the medical record information is uploaded to the blockchain network, and everyone can access the record through an account with specific permissions. Of course, there are still many problems to be solved.

Future work can focus on the following areas for further improvements. Our experiments can be performed on clusters instead of a single PC to optimize the performance of the distributed system. Addressing data storage issues at the fundamental level in blockchain rather than relying solely on FASR technology to mitigate storage pressure. Besides, additional features should be introduced into the framework, such as supporting external institutions to access medical records securely and implementing encrypted retrieval.

Acknowledgments. This work is supported in part by the Top-notch Innovative Talent Training Program for Graduate students of Shanghai Maritime University under Grant 2021YBR008.

References

1. Edward H Shortliffe. The evolution of electronic medical records. *Academic Medicine*, 74(4):414–9, 1999.
2. AL Rector, WA Nowlan, and Shazia Kay. Foundations for an electronic medical record. *Methods of information in medicine*, 30(03):179–186, 1991.
3. Jiahong Cai, Wei Liang, Xiong Li, Kuanching Li, Zhenwen Gui, and Muhammad Khurram Khan. Gtxchain: A secure iot smart blockchain architecture based on graph neural network. *IEEE Internet of Things Journal*, 2023.

4. Dezhi Han, Yujie Zhu, Dun Li, Wei Liang, Alireza Souri, and Kuan-Ching Li. A blockchain-based auditable access control system for private data in service-centric iot environments. *IEEE Transactions on Industrial Informatics*, 18(5):3530–3540, 2021.
5. Elias Drakopoulos and Matt Merges. Performance study of client-server storage systems. In *[1991] Digest of Papers Eleventh IEEE Symposium on Mass Storage Systems*, pages 67–68. IEEE Computer Society, 1991.
6. Shivansh Kumar, Aman Kumar Bharti, and Ruhul Amin. Decentralized secure storage of medical records using blockchain and ipfs: A comparative analysis with future directions. *Security and Privacy*, 4(5):e162, 2021.
7. Wei Liang, Yang Yang, Ce Yang, Yonghua Hu, Songyou Xie, Kuan-Ching Li, and Jiannong Cao. Pdpchain: A consortium blockchain-based privacy protection scheme for personal data. *IEEE Transactions on Reliability*, 2022.
8. Nazanin Zahed Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162:102656, 2020.
9. Hongzhi Li, Dezhi Han, and Mingdong Tang. A privacy-preserving storage scheme for logistics data with assistance of blockchain. *IEEE Internet of Things Journal*, 9(6):4704–4720, 2021.
10. Jiatao Li, Dezhi Han, Zhongdai Wu, Junxiang Wang, Kuan-Ching Li, and Arcangelo Castiglione. A novel system for medical equipment supply chain traceability based on alliance chain and attribute and role access control. *Future Generation Computer Systems*, 142:195–211, 2023.
11. Ed Coyne and Timothy R Weil. Abac and rbac: scalable, flexible, and auditable access management. *IT professional*, 15(03):14–16, 2013.
12. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.
13. Chongqing Chen, Dezhi Han, and Xiang Shen. Clvin: Complete language-vision interaction network for visual question answering. *Knowledge-Based Systems*, page 110706, 2023.
14. Chongqing Chen, Dezhi Han, and Chin-Chen Chang. Caan: Context-aware attention network for visual question answering. *Pattern Recognition*, 132:108980, 2022.
15. Dun Li, Dezhi Han, Zibin Zheng, Tien-Hsiung Weng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. Moocschain: A blockchain-based secure storage and sharing scheme for moocs learning. *Computer Standards & Interfaces*, 81:103597, 2022.
16. Dun Li, Dezhi Han, Tien-Hsiung Weng, Zibin Zheng, Hongzhi Li, Han Liu, Arcangelo Castiglione, and Kuan-Ching Li. Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. *Soft Computing*, 26(9):4423–4440, 2022.
17. Baowei Wang, Shi Jiawei, Weishen Wang, and Peng Zhao. Image copyright protection based on blockchain and zero-watermark. *IEEE Transactions on Network Science and Engineering*, 9(4):2188–2199, 2022.
18. Pengbin Han, Aina Sui, Tao Jiang, and Chaonan Gu. Copyright certificate storage and trading system based on blockchain. In *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pages 611–615. IEEE, 2020.
19. Nan Jing, Qi Liu, and Vijayan Sugumaran. A blockchain-based code copyright management system. *Information Processing & Management*, 58(3):102518, 2021.
20. Chiara Garilli. Blockchain and smart contracts: New perspectives on copyright protection in the digital single market. In *Handbook of Research on Applying Emerging Technologies Across Multiple Disciplines*, pages 159–175. IGI Global, 2022.
21. Lijun Xiao, Weihong Huang, Yong Xie, Weidong Xiao, and Kuan-Ching Li. A blockchain-based traceable ip copyright protection algorithm. *IEEE Access*, 8:49532–49542, 2020.

22. Na Gao, Dezhi Han, Tien-Hsiung Weng, Benhui Xia, Dun Li, Arcangelo Castiglione, and Kuan-Ching Li. Modeling and analysis of port supply chain system based on fabric blockchain. *Computers & Industrial Engineering*, 172:108527, 2022.
23. Bing Qing Tan, Fangfang Wang, Jia Liu, Kai Kang, and Federica Costa. A blockchain-based framework for green logistics in supply chains. *Sustainability*, 12(11):4656, 2020.
24. Ahmad Musamih, Khaled Salah, Raja Jayaraman, Junaid Arshad, Mazin Debe, Yousof Al-Hammadi, and Samer Ellahham. A blockchain-based approach for drug traceability in healthcare supply chain. *IEEE access*, 9:9728–9743, 2021.
25. Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. Blockchain technology and its relationships to sustainable supply chain management. *International journal of production research*, 57(7):2117–2135, 2019.
26. Guang Chen, Bing Xu, Manli Lu, and Nian-Shing Chen. Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*, 5(1):1–10, 2018.
27. Preeti Bhaskar, Chandan Kumar Tiwari, and Amit Joshi. Blockchain in education management: present and future applications. *Interactive Technology and Smart Education*, 18(1):1–17, 2021.
28. Kim Beom Rii. Digital ilearning chain scheme in education blockchain based. *Aptisi Transactions on Technopreneurship (ATT)*, 4(2):174–183, 2022.
29. Hongzhi Li and Dezhi Han. Edurss: A blockchain-based educational records secure storage and sharing scheme. *IEEE access*, 7:179273–179289, 2019.
30. Yi Chen, Shuai Ding, Zheng Xu, Handong Zheng, and Shanlin Yang. Blockchain-based medical records secure storage and medical service framework. *Journal of medical systems*, 43:1–9, 2019.
31. Igor Radanović and Robert Likić. Opportunities for use of blockchain technology in medicine. *Applied health economics and health policy*, 16:583–590, 2018.
32. Jia Qu. Blockchain in medical informatics. *Journal of Industrial Information Integration*, 25:100258, 2022.
33. Hu Liu and Yuxuan Liu. Construction of a medical resource sharing mechanism based on blockchain technology: evidence from the medical resource imbalance of china. In *Healthcare*, volume 9, page 52. MDPI, 2021.
34. Mhamad Bakro, Sukant K Bisoy, Ashok K Patel, and M Adib Naal. Hybrid blockchain-enabled security in cloud storage infrastructure using ecc and aes algorithms. In *Blockchain based Internet of Things*, pages 139–170. Springer, 2022.
35. Yassine El Khanboubi, Mostafa Hanoune, and Mohamed El Ghazouani. A new data deletion scheme for a blockchain-based de-duplication system in the cloud. *Int. J. Commun. Netw. Inf. Secur. IJCNIS*, 13:331–339, 2021.
36. Sisi Zhou, Kuanching Li, Lijun Xiao, Jiahong Cai, Wei Liang, and Arcangelo Castiglione. A systematic review of consensus mechanisms in blockchain. *Mathematics*, 11(10):2248, 2023.
37. Chris Dannen. *Introducing Ethereum and solidity*, volume 1. Springer, 2017.
38. Dejan Vujičić, Dijana Jagodić, and Siniša Ranić. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE, 2018.
39. Eric Yuan and Jin Tong. Attributed based access control (abac) for web services. In *IEEE International Conference on Web Services (ICWS'05)*. IEEE, 2005.
40. David Ferraiolo, Janet Cugini, D Richard Kuhn, et al. Role-based access control (rbac): Features and motivations. In *Proceedings of 11th annual computer security application conference*, pages 241–48, 1995.
41. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full aes-192 and aes-256. In *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*, pages 1–18. Springer, 2009.

42. Shiwen Zhang, Biao Hu, Wei Liang, Kuan-Ching Li, and Brij B Gupta. A caching-based dual k-anonymous location privacy-preserving scheme for edge computing. *IEEE Internet of Things Journal*, 2023.
43. Han Liu, Dezhi Han, Mingming Cui, Kuan-Ching Li, Alireza Souri, and Mohammad Shojafar. Idenmultisig: identity-based decentralized multi-signature in internet of things. *IEEE Transactions on Computational Social Systems*, 2023.
44. Dezhi Han, Nannan Pan, and Kuan-Ching Li. A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection. *IEEE Transactions on Dependable and Secure Computing*, 19(1):316–327, 2020.
45. Wenbin Yao, Mengyao Hao, Yingying Hou, and Xiaoyong Li. Fasr: An efficient feature-aware deduplication method in distributed storage systems. *IEEE Access*, 10:15311–15321, 2022.
46. Marcela T de Oliveira, Yiannis Verginadis, Lúcio HA Reis, Evgenia Psarra, Ioannis Patiniotakis, and Sílvia D Olabarriaga. Ac-abac: Attribute-based access control for electronic medical records during acute care. *Expert Systems with Applications*, 213:119271, 2023.
47. Mingxiao Du, Qijun Chen, Jieying Chen, and Xiaofeng Ma. An optimized consortium blockchain for medical information sharing. *IEEE Transactions on Engineering Management*, 68(6):1677–1689, 2020.
48. Zhijie Sun, Dezhi Han, Dun Li, Xiangsheng Wang, Chin-Chen Chang, and Zhongdai Wu. A blockchain-based secure storage scheme for medical information. *EURASIP Journal on Wireless Communications and Networking*, 2022(1):40, 2022.

Aoao Bian received the B.S. degree in Computer science and Technology from the Anqing Normal University , Anqing, China, in 2022,and he is currently pursuing the M.S. degree in Shanghai Maritime University, Shanghai, China, in 2024..Her main research interests include blockchain technology and its applications and cryptography.

Dezhi Han received the B.S. degree from Hefei University of Technology, Hefei, China, the MS degree and Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China. He is currently a professor of computer science and engineering at Shanghai Maritime University. His specific interests include storagearchitecture, blockchain technology, cloud computing security and cloud storage security technology.

Mingming Cui received a B.S. degree in Computer Science and Technology from the Anhui University of Finance and Economics, China. She is currently pursuing a Ph.D. degree from Shanghai Maritime University, China, and a Visiting Ph.D. student at the Nanyang Technological University, Singapore. Her research interests include cryptology, blockchain, data privacy protection, network security, VANETS security, and the Internet of things.

Dun Li received the B.S. degree in Human Resource Management from the Huaqiao University, Quanzhou, China, in 2013, and the M.S. degree in Finance from the Macau University of Science and Technology, Macau, China, in 2015. He is currently doing his Ph.D. degree in Information Management and Information Systems at Shanghai Maritime University. His main research interests include smart finance, big data, machine learning, IoT, and Blockchain.

Received: October 02, 2023; Accepted: January 25, 2024.