Sun *et al. J Wireless Com Network* (2022) 2022:40
https://doi.org/10.1186/s13638-022-02122-6

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A blockchain-based secure storage scheme for medical information

Zhijie Sun[1], Dezhi Han[1], Dun Li[1,2*], Xiangsheng Wang[1], Chin-Chen Chang[3] and Zhongdai Wu[4]

*Correspondence: lidunshmtu@outlook.com
[1] College of Information Engineering, Shanghai Maritime University, Shanghai, China
Full list of author information is available at the end of the article

Abstract

Medical data involves a large amount of personal information and is highly privacy sensitive. In the age of big data, the increasing informatization of healthcare makes it vital that medical information is stored securely and accurately. However, current medical information is subject to the risk of privacy leakage and difficult to share. To address these issues, this paper proposes a healthcare information security storage solution based on hyperledger fabric and the attribute-based access control framework. The scheme first utilizes attribute-based access control, which allows dynamic and fine-grained access to medical information, and then stores the medical information in the blockchain, which can be secured and tamper-proof by formulating corresponding smart contracts. In addition, this solution also incorporates IPFS technology to relieve the storage pressure of the blockchain. Experiments show that the proposed scheme combining access control of attributes and blockchain technology in this paper can not only ensure the secure storage and integrity of medical information but also has a high throughput when accessing medical information

Keywords: Medical information, Hyperledger fabric, Smart contracts, ABAC, IPFS

1 Introduction

With the development of technology, various emerging technologies are merging with the healthcare sector, making the process of building healthcare information technology increasingly sophisticated [1]. The World Health Organization defines medical information as the most innovative and shareable asset. Nowadays, the number of medical institutions around the world presents an index stage growth, and the medical data generated by medical institutions also presents explosive growth. Due to the deepening of the degree of information in hospital information, the information system within the hospital gradually expands from a single HIS charging system into a system with electronic medical records. The medical data is accompanied by the registration, diagnosis, and hospitalization, medical data is gradually complex and stereochemical, and the importance of privacy and security is significantly increased [2].

Currently, the combination of traditional paper medical records and centralized medical data management systems is still the main form of medical institutions to store patients' medical data as shown in Fig. 1. However, this form of medical system faces severe risks of privacy disclosure [3]. Therefore, the transformation of the centralized

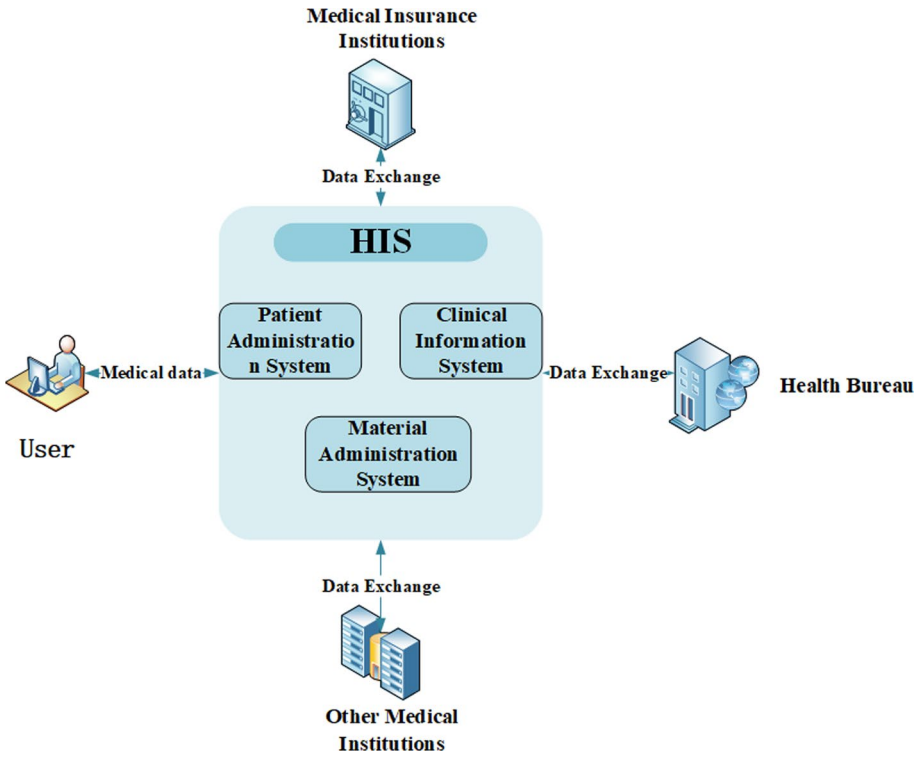


Fig. 1 Medical data exchange

medical data management system to distributed medical data-sharing system is an irresistible trend of the whole society [4].

However, since most medical and health institutions are isolated from each other, they store and maintain medical health data, forming data islands. This is not only not conducive to long-term records of patients with their disease development, but also caused a waste of medical equipment and a large number of medical health data resources duplication. To maximize the value of medical health data, to meet the core needs of medical information construction, and provide more humanized and reasonable services for patients, sharing data between medical institutions is an inevitable trend [5]. In addition, due to the extensive use of emerging Internet technology in the medical field, the medical data transmission methods and paths have become increasingly diversified, and gradually transferred from the internal transmission of hospitals to the transmission between medical institutions, medical institutions, and insurance and other institutions, and between patients and medical institutions, which also greatly increases the difficulty of patient data protection [6]. The above reasons lead to the characteristics of large scale, complex structure, and rapid growth of medical data, so it is difficult to find an ideal method to store medical information.

Fortunately, in recent years, the rise of blockchain technology has brought new solutions to the secure storage of medical information. In essence, blockchain is a distributed database with the characteristics of decentralization, security, and transparency [7–9]. As a decentralized database, blockchain provides a reliable solution to the problems of poor sharing, low effectiveness, and weak security in medical data management. Data

can be recorded on the real-time shared blockchain platform, and timestamps are added to ensure the immutability of the data. The tamper resistance of the blockchain ensures the security of medical data [10]. On the licensed blockchain, blockchain members can obtain data information through access operations.

Specifically, on the license blockchain, the blockchain member can obtain information of data by accessing operations, allowing the member to view outline information, to ensure sharing of medical data on a non-licked blockchain. Mainstream blockchain projects can be divided into four categories: cryptocurrency, platform, application, and asset token. Blockchain technology is widely used in smart cities [11, 12], Internet of things (IoT) [13–16], smart finance [17], Internet of Vehicles (IoV) [18–22], and education [23–25]. Medical data involves personal privacy and sensitive information, such as personal name, ID number, and home address, so medical records become the primary goal of information theft, so it is urgent to combine blockchain technology and the medical sector.

Furthermore, blockchain has entered a new era with the emergence and continuous improvement of smart contracts and further development of blockchain projects such as Ether and Hyperledger. Smart contracts are programmable and Turing-complete [26]. Transactions can automatically initiate code based on rules set by the system, and the emergence of smart contracts has laid an important foundation for merging blockchain technology and medical information [27]. In the open network environment of blockchain, the attribute-based access control (ABAC) model is a suitable and effective access control model. As a flexible fine-grained access control method [28], the model mainly determines that the data requester has the correct attributes to determine the data requester's access control authority to private data resources.

So far, the application of blockchain technology in the medical field is not satisfactory. In this regard, we store medical data into blockchain by deploying intelligent contracts to ensure the privacy and security of medical data. At the same time, the ABAC model is introduced for access control to ensure that users can access them safely and efficiently. In addition, due to the huge and complex medical information, to alleviate the storage pressure of blockchain, we also combine the interstellar file system to realize the slimming of the whole blockchain and further improve the efficiency of user access. Compared with existing studies, the model proposed in this paper realizes more fine-grained access to medical information and at the same time alleviates the storage pressure of blockchain, making the throughput of the system greatly improved, which is also the advantage of this scheme.

Specifically, the main contributions of this study are as follows.

- This paper applies blockchain to medical information management and realizes decentralized management and secure storage with the help of distributed consensus and authentication mechanisms.
- We design an auxiliary architecture based on ABAC, which can realize fine-grained access control and dynamic management of permissions.
- In this paper, we use smart contracts to define multi-tier data structures, access policies, and system workflows to improve the efficiency of data storage, retrieval, and query.

- We ease the storage pressure of blockchain with the interstellar file system.
- This paper designs simulation experiments and verifies the performance of the scheme.

The rest of this article is as follows. Section 2 describes the related works. In Sect. 3, we introduce the necessary background and technologies. Next, Sect. 4 introduces the model, assumptions, and design objectives of the proposed scheme. Then, Sect. 5 sets up two groups of comparative experiments and then analyzes the results. Finally, in Sect. 6, we summarize this paper and discuss further work.

2 Related work

In this section, we survey blockchain-based secure storage in Sect. 2.1 and blockchain-based secure sharing in Sect. 2.2. Although existing models and schemes achieve secure storage and sharing of medical information, they fail to realize fine-grained access to medical information, which will undoubtedly reduce the user experience. In addition, most existing studies have not considered the storage bottleneck of blockchain. In order to make up for the deficiency of existing studies, this paper not only achieves the safe storage and sharing of medical information, but also optimizes the access control operation of medical information, and alleviates the storage pressure of blockchain to a certain extent, which is also the difference between the proposed scheme in this paper and the existing model.

2.1 Blockchain-based secure storage of medical data

The extension of blockchain technology to the healthcare field has a profound impact due to its decentralized, tamper-proof, and transparent nature.

Azaria et al. [29] propose a decentralized blockchain-based MedRec system to handle EHR. MedRec has a modular design where the administrative privileges, authorization, and data sharing of the system are among the participants. Medblock [30] is a hybrid architecture based on blockchain to protect EMR. The architecture nodes of the architecture are divided into endorsement nodes, sorting nodes, and submission nodes. The consensus algorithm used is a variant of the part consensus algorithm. Conceição et al. [31] propose a generic architecture for storing patient Electronic Health Record (EHR) data using blockchain technology. Yang and Li [32] propose an EHR architecture based on blockchain. The architecture prevents tampering and abuse of EHR by tracking all events in the blockchain. Kushch et al. [33] proposed a special data structure for storing electronic medical data on the blockchain: blockchain tree. The structure of the blockchain tree is a sub-chain and one or more of a recorded patient identity and a sub-chain stored in additional critical information (such as diagnostic records), and blocks on the main chain are initial blocks of the sub-chain.

2.2 Blockchain-based secure sharing of medical data

In addition to safe storage, the blockchain is equally widely used in security sharing. In medical record management, the application and research of the blockchain in the medical field have received much attention, and many research institutions around the world participate.

Xia et al. [34] proposed a blockchain-based system called men shared. The system can minimize the risk of data privacy and can be used to solve the problem of medical data sharing between medical data custodians in an untrusted environment. Zhang et al. [35] propose a blockchain-based medical data-sharing scheme, which uses the private blockchain owned by the hospital to store the patient's health data and uses the consortium blockchain to save the security index. Zhang et al. [36] combined with artificial intelligence technology and blockchain technology proposed a safe and transparent medical data-sharing platform. This platform utilizes the transparency of the zone chain for data tracking, imparting the characteristics of non-tampered. Liu et al. [37] use blockchain technology and cloud storage technology to propose a data-sharing scheme for paying attention to privacy protection in the medical field. The scheme stores the original medical data in the cloud indexes the data in the blockchain and prevents the data from being maliciously modified by the tamper-proof feature of the blockchain. To realize the dynamic communication between medical alliance chains, Qiao et al. [38] propose a scheme that allows dynamic communication between healthcare alliance chains, which enables patients to securely and autonomously share their records in an authorized healthcare alliance chain within milliseconds.

3 Preliminaries

This section mainly introduces the architecture of medical information security storage schemes based on blockchains and access controls. Section 3.1 introduces the structure of the scheme, Sect. 3.2 presents the workflow, and Sects. 3.3 and 3.4 describe the smart contract design.

3.1 Blockchain technology in healthcare information storage

Blockchain helps to build decentralized data-sharing and application mechanisms. Traditionally, medical information management is a unilaterally maintained information system. The drawback of this mode of management is that too centralized information management power makes it difficult to achieve real information sharing. Blockchain technology introduces the characteristics of distributed books. Since the file information input under the blockchain technology is jointly maintained and supervised by multiple parties, the joint supervision of various information data by multiple departments ensures the openness and transparency of data information and also determines the openness and transparency of blockchain technology transactions rules [39]. This will fundamentally solve the problems of low work efficiency and too chaotic a working state in traditional medical information management.

Moreover, blockchain can construct a credible deposit system. The management of medical archives information is nothing more than the four most basic processes of addition, deletion, modification, and query. However, in blockchain, the two basic processes of deletion and modification in archives information management are abandoned, and the process of archives information processing is reduced. The irreparability and security of data information in blockchain are guaranteed from the technical design. In addition, each block of information in the blockchain records the creation time and the hash value of the previous block. This chain structure marked with time itself facilitates

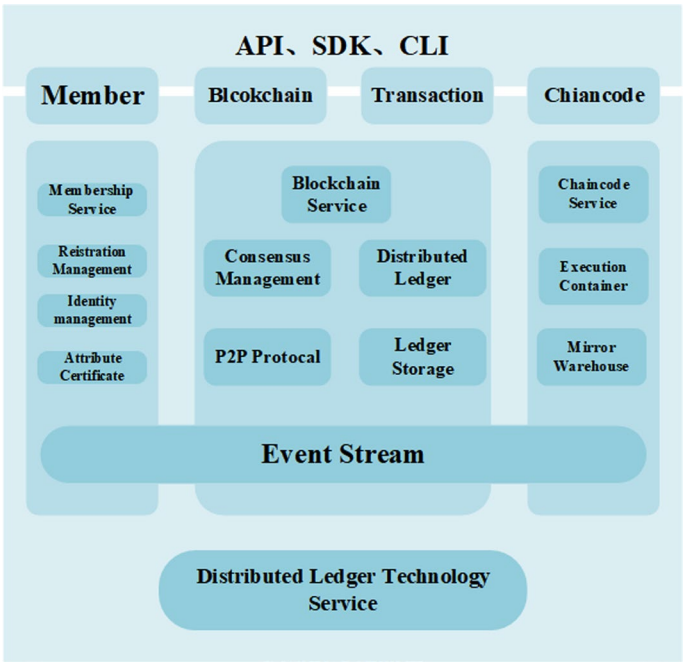


Fig. 2 The module architecture diagram of hyperledger fabric

the usual audit, tracking, and traceability, and improves the utilization rate of medical information.

Finally, blockchain can solidify data exchange and benefit allocation rules. The combination of intelligent contract and block link technology can maximize the automation of archival information sharing. Once the smart contract is implemented, it cannot stop and is not interfered with by external operations. Hospitals can use this feature to entrench interest distribution rules [40]. In medical information sharing, intelligent contracts can change the behavior of participants involved in information sharing into active participation, promote the efficiency and speed of information sharing, and truly maximize the value of medical information. In this compulsory information sharing, the secret box operation in traditional information sharing is constrained, and the quality of medical data information is ensured.

3.2 Hyperledger fabric

In recent years, cryptocurrencies, represented by Bitcoin, have achieved great success, which has successfully drawn the world's attention to blockchain technology; however, such public chains have problems such as low transaction throughput, long transaction times, wasted resources, and data consistency. To address these issues, the Linux Foundation created the Hyperledger project in 2015, which is one of the world's largest blockchain projects and is often used as a platform for enterprise blockchain development. Hyperledger fabric is designed with a modular architecture that includes members, blockchain, transactions, and smart contracts, as shown in Fig. 2.

Member management module strengthens the user's joining rights according to the security and privacy requirements of the enterprise-level blockchain. Anyone

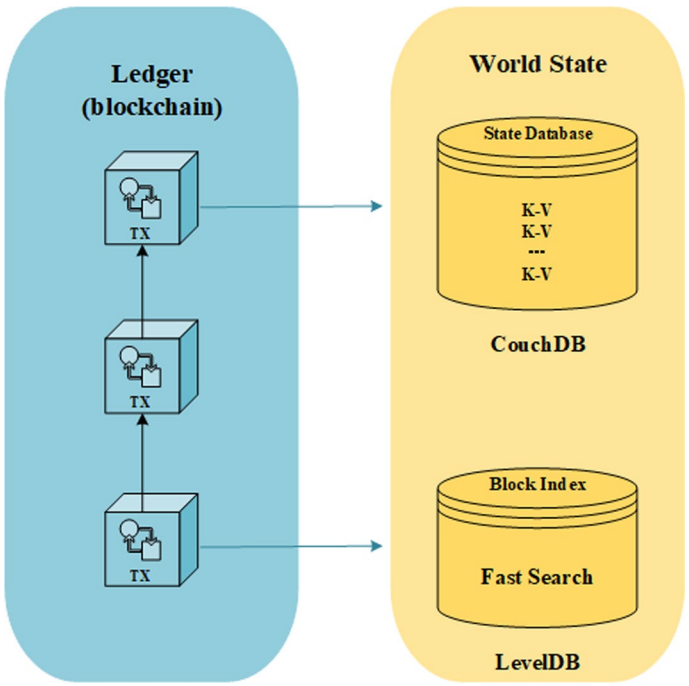


Fig. 3 Structure of the ledger in hyperledger fabric

participating in the transaction needs to be authenticated by the PKI public key infrastructure to join. The blockchain module uses the P2P protocol to manage the distributed ledger. Different consensus protocols can be configured according to different needs. The chain ledger records the transaction history, and the World State mechanism records the latest state of the ledger, as shown in Fig. 3. Hyperledger fabric employs Apache Kafka (Distributed Messaging System) based on ZooKeeper (Distributed Services Framework). Kafka is essentially a message processing system where consumers of messages subscribe to specific topics and producers are responsible for publishing messages. In the whole hyperledger fabric network, KafKa mainly provides transaction ordering service; that is, KafKa realizes the ordering service for all transaction requests in the network.

The transaction module controls the data in the transaction process in the form of deployment transactions and invocation transactions, where deployment transactions are installed on all peer nodes by Chaincode when the transaction is successfully executed, while invocation transactions are conducted by invoking the specified functions in the Chaincode through the SDK provided by the Fabric Software Development Kit. Smart contracts record the business logic agreed by members of Fabric's federated chain and can be written in common languages such as Go and Java, overcoming the shortcomings of traditional blockchains that are limited to domain-specific languages.

3.3 Attribute-based access control model

Attribute-based access control is a comprehensive consideration of user, resource, operation, and contextual access control policies. It determines whether to grant access to the requester to configure the correct attribute; that is, this policy does not need to

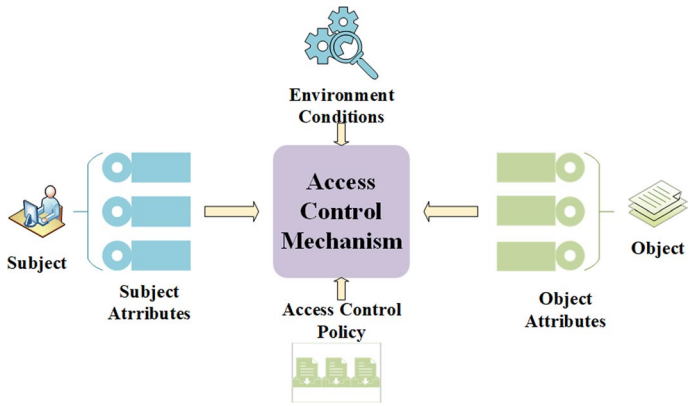


Fig. 4 Structure of the ABAC model

specify the relationship between the data requester and the private data, but by judging whether the data requester's attribute determines its pair access control permissions for this private data. As the system is running, the strategy is a relatively stable attribute. Therefore, using attributes to describe access control policies can not only separate attribute management and access decision-making, but also refine access control granularity, which has good flexibility and scalability, and is more conducive to adding, deleting, update and query policies. Attributes are the core of the policy, which can be defined by a quadruplet $A \in \{S, O, P, E\}$, where each field has the following meaning: A represents attributes, each of which exists as a key-value pair. S represents subject attributes, including the subject's identity, role, position, and credentials. O represents object attributes, including the object's identity, location, department, type, data structure, etc. E represents the environmental attributes, including time, system status, security level, current access, etc. P represents the operation attributes, mainly used to describe the subject's access to the object type, such as write, modify, and delete. The structure of the model is shown in Fig. 4. An attribute-based access control request (ABACR) can be defined as $\text{ABACR} = \{\text{AS} \wedge \text{AO} \wedge \text{AP} \wedge \text{AE}\}$, where AS represents the subject attribute, AO represents the object attribute, AP represents the operation attribute, and AE represents the environment attribute. R represents a set of rules, which can also be defined by a quadruplet: $R(A(S_i), A(O_i), A(E_i), A(P_i)) \rightarrow \{\text{Allow,Deny}\}$; this formula indicates that the subject authorizing attribute S_i at the time of access performs an access action P_i on object O_i in a contextual environment with attribute value E_i, with two outcomes, i.e., allow or deny.

3.4 Interplanetary file system (IPFS)

Designed by Juan Benet and developed by Protocol Labs with the help of the open-source community since 2014, IPFS (Interstellar File System) is a network transfer protocol designed to create persistent and distributed storage and sharing of files. IPFS combines features of existing technologies, including DHT, BitTorrent, Git, and SFS, to achieve the primary function of storing data locally and connecting nodes to each other for data transfer. IPFS was originally designed to build a better resource network than the now commonly used HTTP protocol to compensate for the shortcomings of HTTP.

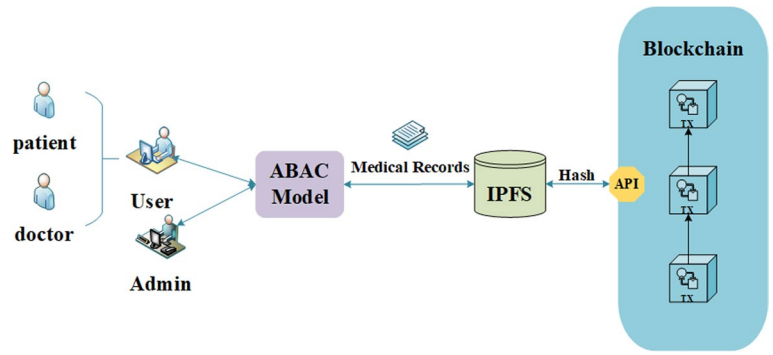


Fig. 5 The proposed scheme architecture

Compared to HTTP, IPFS exhibits advantages such as fast download speeds, global storage, security, and data perpetuation. IPFS is essentially a content-addressable, versioned, peer-to-peer hypermedia distributed storage, and transport protocol. It has the following features. Content Addressable: IPFS only cares about the content of the file, generating a unique hash mark from the file content, which is accessed by the unique mark and checked in advance to see if the mark has already been stored. If it has been stored, it is read directly from other nodes, without the need for duplicate storage, saving space in a sense. Slicing large files: Files placed in IPFS nodes do not care about their storage path or name. IPFS provides the ability to slice and dice large files, downloading multiple slices in parallel when used. Decentralized, distributed network structure: Such a network structure is suitable for solving bottlenecks in the blockchain's storage capacity by storing large amounts of hypermedia data on IPFS. Encrypted storage: IPFS adds a cryptographic hash unique to digital information to the encrypted data, and the corresponding hash of the stored file cannot be changed. The hash corresponds to the file one-to-one. In an IPFS network, there is no need to take into account the location of the server and the name and path of the file. When a file is placed in an IPFS node, each file is given a unique hash value calculated based on its contents. When access to a file is requested, IPFS finds the node where the file is located based on the hash table and fetches the file. IPFS combined with blockchain can be a good solution to the blockchain storage problem.

4 Experimental methods

This section mainly introduces the architecture of medical information security storage schemes based on blockchains and access controls. Section 4.1 introduces the structure of the scheme, Sect. 4.2 presents the workflow, and Sect. 4.3 describes the smart contract design of the scheme.

4.1 Scheme architecture

The architecture of the system consists of a user, an attribute-based access control model, an interstellar file system, and a blockchain, the detailed architecture of which is shown in Fig. 5.

Users can be divided into two types: normal users, who can be doctors and patients, both of whom can participate in the authorization of the solution and thus access medical data. Administrator users are responsible for managing the blockchain and can create or update smart contracts. The combination of the ABAC model and this scheme forms a medical information access control model. The specific description of the model is as follows.

$$P = \{\text{AS, AO, AP, AE}\} \tag{1}$$

$$AS = \{\text{userId, role, department}\} \tag{2}$$

$$AO = \{\text{recordId}\} \tag{3}$$

$$\text{AP} = \begin{cases} 1, \text{ allow} \\ 0, \text{ deny} \end{cases} \tag{4}$$

$$\text{AE} = \{\text{createTime, endTime}\} \tag{5}$$

Policy (P): It represents the access control policy based on attributes contains four elements in the set, namely AS, AO, AP, and AE.

Attribute of subject (AS): It includes three main types of attributes, namely user ID (identifies the unique identity of the user), user role (doctor and patient), and user department (specific department).

Attribute of object (AO): It includes the medical record ID (identifies the uniqueness of the record).

Attribute of permission (AP): An attribute that indicates whether the user has access to the medical record, with 1 representing permission and 0 representing denial.

Attribute of environment (AE): The environmental conditions required for the access control policy, mainly including the creation time (when the policy was created) and the end time (when the policy expires). If the current time of a policy is later than the end time, it means that the policy is invalid.

IPFS: It is mainly used to mitigate the storage pressure of the blockchain. The medical data stored in IPFS will be stored in a Merkle DAG to ensure the security of the data, which is called the address hash. Then, the address hash is stored into the zone chain, thereby replacing the original data. In IPFS, the original data is subjected to the SHA256 algorithm twice and then Base58 encoding, resulting in a hash length of 33 Bytes. So the original medical information is replaced with the hash address, which will greatly reduce a block. The size of the whole blockchain is achieved.

Blockchain: The blockchain is the heart of the solution, a distributed network of trusted nodes that ensures the synchronization and storage of medical data, thus ensuring data security and accuracy. In this solution, the blockchain is developed based on hyperledger fabric and access control can be implemented by writing smart contracts.

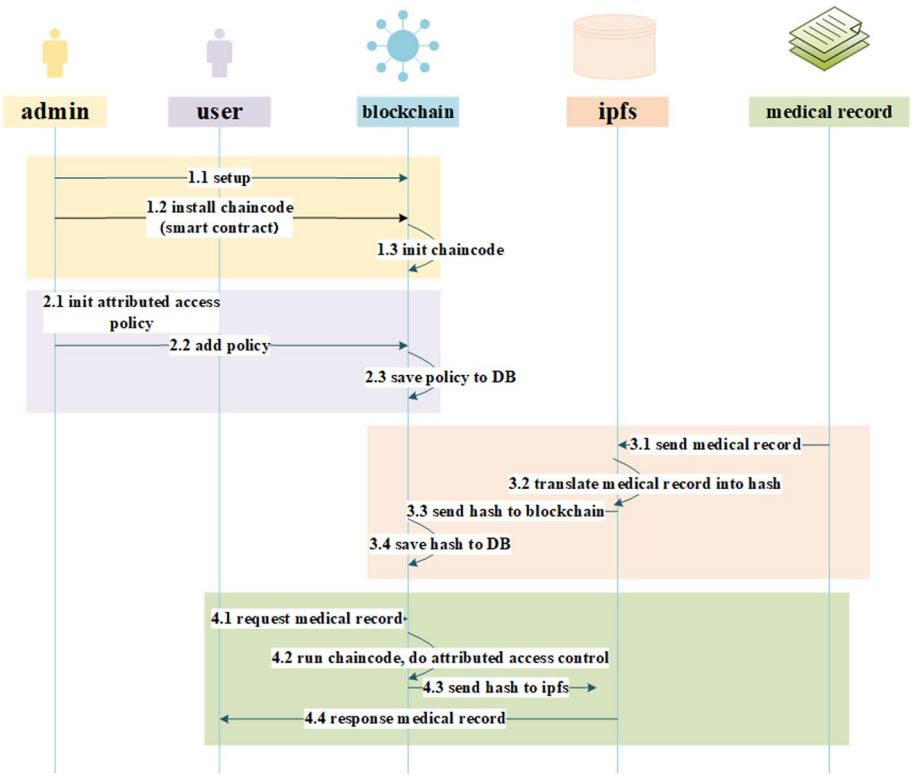


Fig. 6 The workflow of the proposed scheme

4.2 Workflow

The workflow of the proposed scheme mainly contains four parts. This section describes each part, and the specific workflow is shown in Fig. 6. The symbols used are shown in Table 1.

4.2.1 Part 1

The basic procedure of this program is the installation of the construction and Chaincode of the blockchain network. These basic processes need to be completed by the administrator user. Process 1 is mainly divided into three steps as follows.

Step 1: Prior to building a specific blockchain network, all members of the network must register the certificate and the required certificate is issued by CA.

$$\mathrm{CA} \rightarrow \left\{\mathrm{Cert}_{\mathrm{peer}}, \mathrm{Cert}_{\mathrm{orderer}}, \mathrm{Cert}_{\mathrm{channel}}, \mathrm{Cert}_{\mathrm{user}}\right\} \tag{6}$$

All peer nodes and orderer nodes run in Docker containers and the relevant certificates they require need to be packaged into a Docker image before they can be run.

$$\mathrm{Build}(\mathrm{conf}, \mathrm{Cert}) \xrightarrow{\mathrm{build}} \mathrm{Image} \xrightarrow{\mathrm{run}} \mathrm{Container} \tag{7}$$

After setting up all the peer and orderer nodes, start creating channels, each in a separate blockchain and ledger as

Table 1 The summary of acronyms and definitions

Notations	Description
CA	Certificate authority
Cert	Certificate file
conf	Config file of the node
F(x)...	Functions defined in source code
CC	Chaincode in hyperledger fabric
ASC	Access smart contract
PSC	Policy smart contract
RSC	Record smart contract
Image	Docker image
Container	Docker container
TX	Transaction in blockchain
AS, AO, AP, AE	Attributes of subject, object, permission, and environment
Ledger	Ledger in hyperledger fabric
SDB	State database in hyperledger fabric
IPFS	Interplanetary file system
Cli	Blockchain system client
ABACP	Attribute-based access control policy

$$\{\text{blockchain, ledger}\} \xrightarrow{\text{join}} \text{Channel} \tag{8}$$

Step 2: After the above operation, a basic blockchain network has been built, and the Chaincode is written next to create an application.

$$\text{Code}(Fx\ldots) \rightarrow \text{CC} \tag{9}$$

The administrator user uses the hyperledger fabric SDK or Client to install the Chaincode, and all peer nodes must have the Chaincode installed.

$$\text{Install(CC)} \xrightarrow{\text{SDK/Client}} \text{Peer} \tag{10}$$

Step 3: Once the Chaincode is completed, we need to initialize it by calling the invoke function to complete the initialization of the Chaincode, and the initialized Chaincode is stored in the container.

$$\text{Invoke(Init)} \xrightarrow{\text{SDK/Client}} \text{Peer} \tag{11}$$

4.2.2 Part 2

This section requires the specification of relevant access control policies and the whole process needs to be agreed upon between the user and the administrator. The policy needs to be saved to the blockchain by the administrator once it has been created.

Step 1: Administrators and users set access control policies based on AS, AO, AP, and AE.

$$\text{Decide(AS, AO, AP, AE)} \rightarrow \text{ABACP} \tag{12}$$

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

$$\text{Upload(ABACP)} \rightarrow \text{Contract} \tag{13}$$

Step 3: The administrator runs PSC to implement operations such as adding and modifying policies and saves the final policy values to the SDB and ledger.

$$\text{PSC(ABACP)} \rightarrow \{\text{SDB, ledger }\} \tag{14}$$

4.2.3 Part 3

This section implements the storage of medical information by first uploading the medical records into IPFS to get a hash address, and then saving that address to the blockchain.

Step 1: Users upload medical records to IPFS.

$$\text{Upload(Medical Record)} \rightarrow \text{IPFS} \tag{15}$$

Step 2: IPFS translates medical records into a hash address according to its operational mechanism.

$$\text{IPFS(Medical Record)} \xrightarrow{\text{translate}} \text{hash} \tag{16}$$

Step 3: Send the hash address to the blockchain.

$$\text{Send(hash)} \rightarrow \text{blockchain} \tag{17}$$

Step 4: Save medical information to the SDB and ledger by running the smart contract RSC.

$$\text{Run(RSC)} \rightarrow \{\text{SDB, ledger}\} \tag{18}$$

4.2.4 Part 4

This section is a process for accessing medical information based on attribute access control and can be divided into four specific steps.

Step 1: The user initiates a request for access to medical data.

$$\text{Request} \rightarrow \text{blockchain} \tag{19}$$

Step 2: Upon receipt of a user request, the ASC contract is called to verify that the user has access to the data.

$$\text{ASC(Request)} \rightarrow \begin{cases} 1, \text{ allow} \\ 0, \text{ deny} \end{cases} \tag{20}$$

Step 3: If the user has access rights, then the blockchain transfers the hash of the medical information to the IPFS.

$$\text{Blockchain (hash)} \xrightarrow{\text{send}} \text{IPFS} \tag{21}$$

Step 4: IPFS calculates the medical data requested by the user based on the hash address.

$$\text{Response(Medical Record)} \rightarrow \text{Cli} \tag{22}$$

4.3 Smart contract of the scheme

Smart contracts are not only related to the implementation of access control, but also the storage of medical information, and are therefore at the heart of this solution. There are three smart contracts in total: policy contract (PSC), access control contract (ASC), and medical record contract (RSC).

4.3.1 Policy contract (PSC)

The PSC provides the following methods to manipulate ABACPs.

CheckPolicy(): PSC needs to verify the validity of the ABACP by this method. Each ABACP should contain AS, AO, AP, and AE, and all the four attributes should be satisfied for this policy to be valid.

AddPolicy(): The PSC needs to run the *CheckPolicy*() method before calling this method to add the policy, and only after the policy is legal can the policy be written to SDB and blockchain. The details are shown in Algorithm 1.

DeletePolicy(): This method will be called in two ways. Firstly, the administrator will call this method to delete an ABACP. Secondly, when the *CheckAccess*() method is executed and a policy is found to have expired, then this method will be called automatically to delete the useless policy. This is shown in Algorithm 2.

UpdatePolicy(): This method is called when an administrator needs to modify an ABACP. This method is called when the administrator needs to modify an ABACP. The modification record is also written to the SDB and the blockchain. This method also executes the *AddPolicy*() method at the end after the policy is updated, adding the modified policy back to the blockchain.

QueryPolicy(): all policies are stored in the state database CouchDB (a kind of key-value pair database) and the administrator can query the details of the desired ABACP by using the property AS or AO.

Algorithm 1 PSC.AddPolicy()

Require: ABACP
Ensure: Ok or Error
1: APIstub ChaincodeStub ← Invoke()
2: **if** CheckPolicy(ABACP) == False **then**
3: **return** Error(BadPolicy)
4: **end if**
5: AS, AO ← ABACP
6: ABACP_{id} ← HASH_{sha256}(AS + AO)
7: err ← A APIstub.PutState(ABACP_{id}, ABACP)
8: **if** err! = null **then**
9: **return** Error
10: **end if**
11: **return** Ok

Algorithm 2 PSC.DeletePolicy()

Require: AS, AO
Ensure: Ok or Error
1: APIstubChaincodeStub ← Invoke()
2: PolicyID ← HASH_{sha256}(AS + AO)
3: err ← APIstub.GetState(Id)
4: **if** err! = null **then**
5: **return** Error
6: **end if**
7: err ← APIstub.DelState(PolicyID)
8: **if** err! = null then **then**
9: **return** Error
10: **end if**
11: **return** Ok

4.3.2 Access control contract (ASC)

ASC primarily implements the access control function, i.e., determining whether a user's access control-based request matches the prescribed access control policy. The methods in ASC are as follows. *CheckAccess*(): This method is the core of the implementation of access control, as shown in Algorithm 3. If the method returns a null result, it proves that there is no policy that supports the request and the request is invalid. If the result is not null, it means that there is a policy that matches the request. Finally, the request is verified by validating the eligible policy, and if the attributes AE and AP in the policy are both satisfied, the request is proven to pass the verification.

Algorithm 3 ASC.CheckAccess()

Require: ABAC_Request
Ensure: Ok or Error
1: A_uS, A_uO, A_uE ← GetAttrs(ABAC_Request)
2: P ← PSC.QueryPolicy(A_uS, A_uO)
3: **if** P == Null **then**
4: **return** Error();
5: **end if**
6: { . . . , A_pP, A_pE} ← P
7: **if** Value(A_pP) == 1 && A_pE.endTime >currentTime **then**
8: **return** Ok;
9: **end if**
10: **return** Error()

4.3.3 Policy contract (PSC)

The RSC is primarily used to store a hash address representing a complete medical record. The user first uploads the medical record to IPFS, which then returns a hash address for the medical record, which is then uploaded to the blockchain and SDB. *AddRecord*(): This method stores the hash address from IPFS into the blockchain, i.e., the key-value pair $< recordId, hash >$ into the SDB. The details are shown in Algorithm 4.

Table 2 Hardware and software resources

Hardware	
CPU	i7 7500u 2.9 GHz
Memory	8G
Hard disk	1T
Software	
OS	Ubuntu Linux 16.04
Docker	v19.03.2
Docker-compose	v1.24.1
Node	v12.12.0
Golang	v1.15.8
Hyperledger fabric	v1.4.6

Algorithm 4 RSC.AddRecord()

Require: Medical Record(MR)
Ensure: Ok or Error
1: APIstubChaincodeStub ← Invoke()
2: IPFS_{hash} ← IPFS(MR)
3: RecordID ← HASH_{sha256}(MR.recordId)
4: err ← APIstub.PutState(RecordID, IPFS_{hash})
5: **if** err! = null **then**
6: **return** Error
7: **end if**
8: **return** Ok

DeleteRecord(): This method first deletes the hash address from the SDB, and then deletes the complete medical record from the IPFS based on *recordId*. *UpdateRecord*(): When this method is executed, it first updates the medical data in the IPFS to get a new hash address, and then restores this new hash address to the SDB by calling the *AddRecord*() method. *QueryRecord*(): This method first looks up the hash address of the medical record in the SDB based on the *recordId* and then sends the found hash address to IPFS to be parsed into a complete medical record.

5 Experiment and results

This section introduces the process of the experiment and the final results, which are used to verify the performance of this solution through comparison. Section 5.1 introduces the experimental environment, i.e., the hardware and software resources required for the experiment. Section 5.2 introduces the process of creating and implementing the solution. Section 5.3 presents the experimental results, which are used to compare and analyze the performance of the solution.

5.1 Experimental environment

The hardware and software resources required for the stand-alone environment for this solution are shown in Table 2.

5.2 Creation and realization

This section mainly includes three parts: Sect. 5.2.1 mainly introduces the network structure of the scheme and initialization configuration and start. Section 5.2.2 introduces the installation of the Chaincode. Section 5.2.3 mainly introduces how to use the attribute-based access control model to call intelligent contracts.

5.2.1 Network architecture and initialization process

The scheme consists of a total of eight network nodes, and the steps for network initialization are shown below.

Step 1: Use cryptogen tools to generate organization structure and identity certificates for your network.

Step 2: Use the configtxgen tool to generate the creation block for Orderer, the configuration transaction file for the channel, and the anchor node configuration update file for each organization.

Step 3: First start the fabrics network with Docker-compose, then use client nodes to create channels, and finally add each peer node to the channels.

5.2.2 Chaincode installation and upgrade

Firstly, installation. After the initialization of the blockchain network, the Chaincode can be installed. The Chaincode is installed through the hyperledger client node. The client node is used to install the Chaincode into each peer node in turn. Secondly, instantiation. After installing the Chaincode, specify any peer node to instantiate the installed chain code. Finally, upgrade. Before updating the chain code, you must install the new chain code; that is, the chain code update is only valid on the peer node with the new chain code installed.

5.2.3 System implementation

In hyperledger fabric, users can access the blockchain via a client or an SDK; in this scenario, a client written by the SDK will be used to interact with the blockchain. The specific steps are as follows.

Step 1: The CA node generates a key pair for the client, which is stored in the user's wallet.

Step 2: The administrator connects the client to the peer node, and once the link is complete, the transaction can be submitted or evaluated.

Step 3: First the orderer node completes the sorting process, then a consensus is reached between the peer nodes, and finally the status database can be queried or updated. If you want to add a policy, you can call the *AddPolicy*() method in PSC, as shown in Fig. 7.

As shown in Fig. 8, if you want to know whether a policy has been added successfully, you can call the *QueryPolicy*() method in the PSC to query the details of a policy.

As shown in Fig. 9, this policy can be updated by calling the *UpdatePolicy*() method in the PSC for some reason to adapt to the new case.

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js psc AddPolicy '{"AS":{"userId":"13800010002","role":"doctor","d
epartment":"internal medicine"},"AO":{"recordId":"D100010001"},"AP":1,"AE":{"cre
atedTime":1875458182,"endTime":1876468182}}'
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
psc AddPolicy {"AS":{"userId":"13800010002","role":"doctor","department":"intern
al medicine"},"AO":{"recordId":"D100010001"},"AP":1,"AE":{"createdTime":18754581
82,"endTime":1876468182}}
Transaction has been submit, result is: OK
```

Fig. 7 The result of calling the *PSC.AddPolicy*() method

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js psc QueryPolicy 13800010002
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
psc QueryPolicy 13800010002
Transaction has been submit, result is: {"AE":{"createdTime":1875458182,"endTime
":1876468182},"AO":{"recordId":"D100010001"},"AP":1,"AS":{"department":"internal
 medicine","role":"doctor","userId":"13800010002"}}
```

Fig. 8 Results of calling the *PSC.QueryPolicy*() method

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js psc UpdatePolicy '{"AS":{"userId":"13800010002","role":"patient
","department":"surgery"},"AO":{"recordId":"D100010001"},"AP":1,"AE":{"createdTi
me":1875458182,"endTime":1876468182}}'
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
psc UpdatePolicy {"AS":{"userId":"13800010002","role":"patient","department":"su
rgery"},"AO":{"recordId":"D100010001"},"AP":1,"AE":{"createdTime":1875458182,"en
dTime":1876468182}}
Transaction has been submit, result is: OK
```

Fig. 9 Results of calling the *PSC.UpdatePolicy*() method

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js psc DeletePolicy 13800010002
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
psc DeletePolicy 13800010002
Transaction has been submit, result is: OK
```

Fig. 10 Results of calling the *PSC.DeletePolicy*() method

If a policy becomes invalid or the administrator needs to force the deletion of a policy, the policy can be deleted by calling the *DeletePolicy*() method in the PSC. This is shown in Fig. 10.

As shown in Fig. 11, if the Medical Centre needs to add a new medical record, it can do so by calling the *AddRecord*() method in the RSC. As shown in Fig. 12, if the medical center needs to query the details of a medical record, it can do so by calling the *QueryRecord*() method in the RSC. If a medical record needs to be adjusted in real time due to a new change in the patient's condition, the *UpdateRecord*() method in the RSC can be called to update a medical record. If a medical record needs to be deleted due to age or other reasons, it can be deleted by calling the *DeleteRecord*() method in the RSC.

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js rsc AddRecord '{"recordId":"R19787977983","doctorId":"D13687897
4908","patientId":"13800010002","patientName":"jack","patientAge":20,"patientBir
thday":"1997-09-17","patientPhone":"18502878176","patientAddress":"shanghai","do
ctorName":"merry","diagnosisResult":"fever","diagnosisTime":"2021-07-12","diagno
sisDapartment":"internal medicine"}'
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
rsc AddRecord {"recordId":"R19787977983","doctorId":"D136878974908","patientId":
"13800010002","patientName":"jack","patientAge":20,"patientBirthday":"1997-09-17
","patientPhone":"18502878176","patientAddress":"shanghai","doctorName":"merry",
"diagnosisResult":"fever","diagnosisTime":"2021-07-12","diagnosisDapartment":"in
ternal medicine"}
Transaction has been submit, result is: OK
```

Fig. 11 Results of calling the *RSC.AddRecord*() method

After receiving the user's request, it will automatically call the *CheckAccess*() method in ASC to verify whether the request is reasonable. If the request is reasonable, it will return the 'valid request !'; otherwise, the request is invalid. The details are shown in Fig. 13.

5.3 Results and discussion

To verify the performance of this scheme, two sets of experiments are designed. In the first set of experiments, the number of virtual concurrent clients is set to 200, 400, 600, 800, and 1000, showing the time and throughput of PSC, ASC, and RSC in dealing with transactions under different concurrent requests. The details are shown in Figs. 14 and 15.

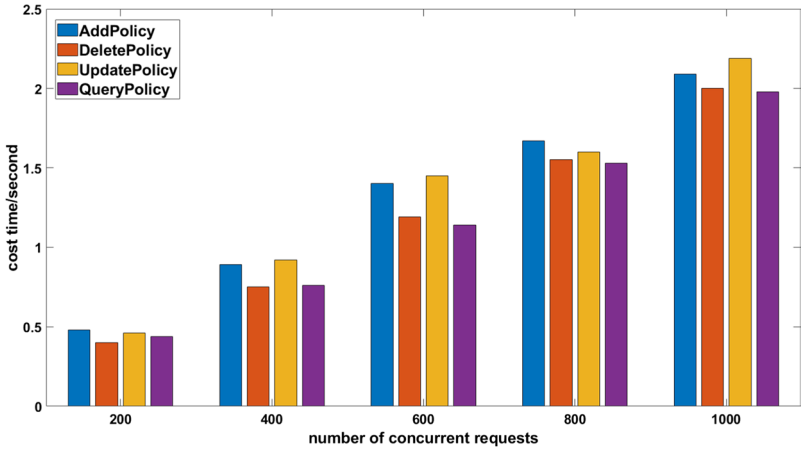
The following conclusions are drawn from the above experimental results: Firstly, add and update operations take a longer time, while query and delete operations take less time. Secondly, the throughput of add and update operations is less than that of query and delete operations. The throughput does not decrease significantly when the number of concurrent requests reaches a certain value. Although PoW consensus can achieve complete decentralization, taking too long to reach consensus results in a large waste of

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js rsc QueryRecord R19787977983
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
rsc QueryRecord R19787977983
Transaction has been submit, result is: {"DiagnosisDapartment":"internal medicin
e","DiagnosisResult":"fever","DiagnosisTime":"2021-07-12","DoctorId":"D136878974
908","DoctorName":"merry","PatientAddress":"shanghai","PatientAge":20,"PatientBi
rthday":"1997-09-17","PatientPhone":"18502878176","RecordId":"R19787977983","pat
ientId":"13800010002","patientName":"jack"}
```

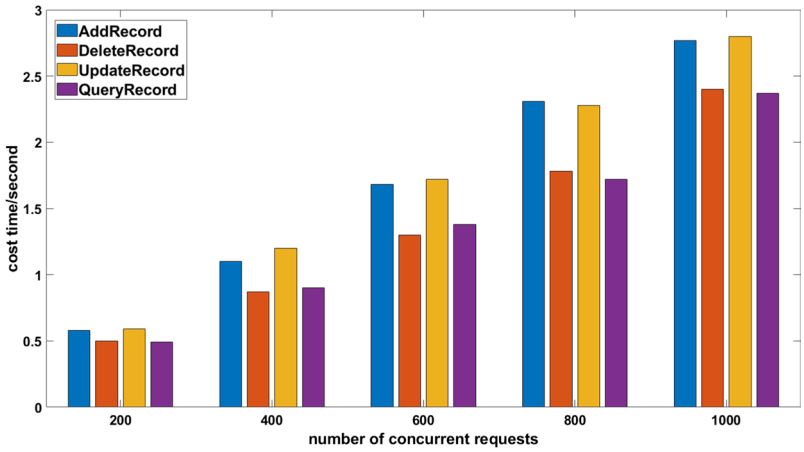
Fig. 12 The result of calling the *RSC.QueryRecord*() method

```
szj@szj-virtual-machine:~/goworkstation/src/github.com/medical.com/client/nodejs
$ node invoke.js asc CheckAccess '{"AS":{"userId":"13800010002","role":"doctor",
"department":"internal medicine"},"AO":{"recordId":"D100010001"}}'
Wallet path: /home/szj/goworkstation/src/github.com/medical.com/client/nodejs/wa
llet
asc CheckAccess {"AS":{"userId":"13800010002","role":"doctor","department":"inte
rnal medicine"},"AO":{"recordId":"D100010001"}}
Transaction has been submit, result is: valid request!
```

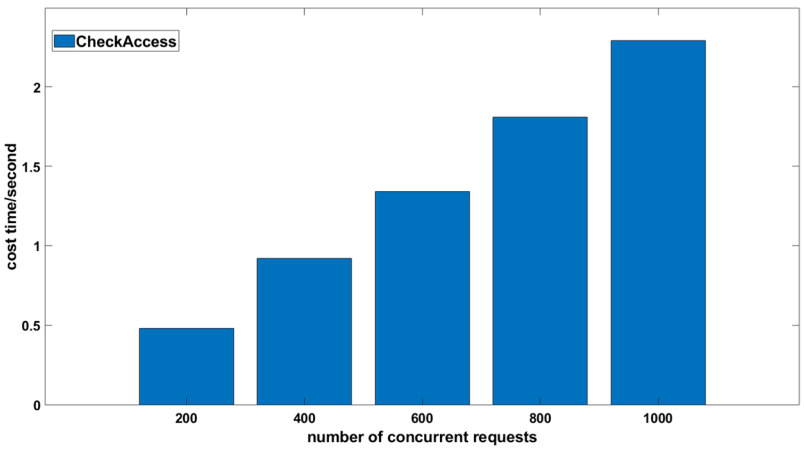
Fig. 13 Results of calling *ASC.CheckAccess*() method



(a) Time spent in PSC with different concurrent requests

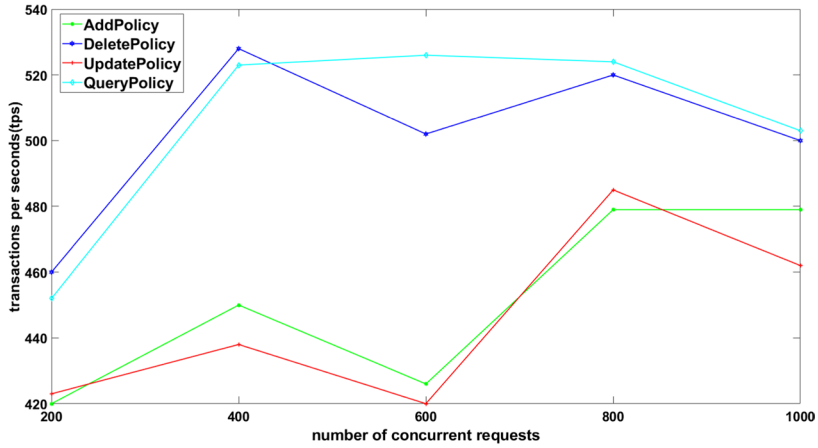


(b) Time spent by RSC under different concurrent requests

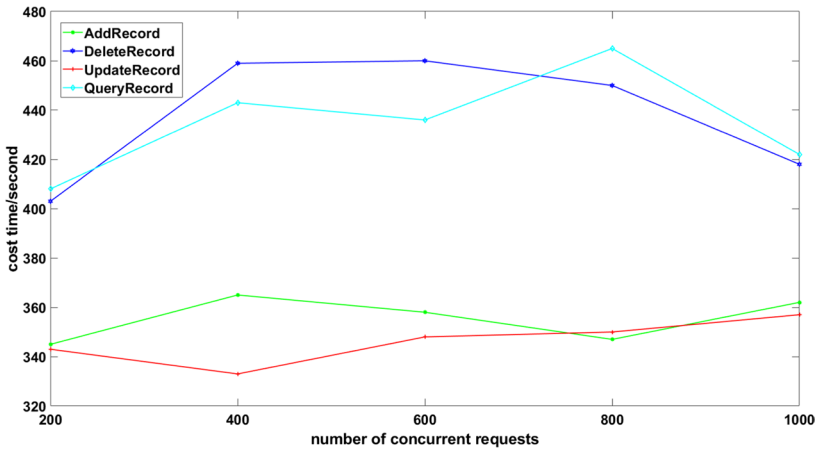


(c) Time spent by ASC under different concurrent requests

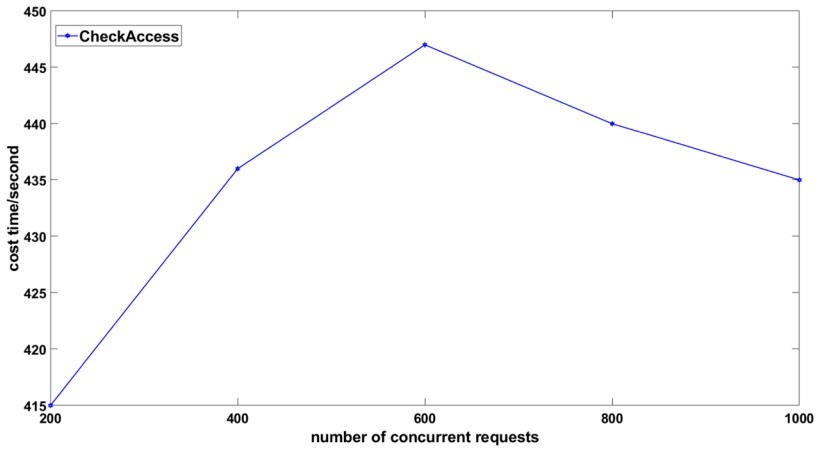
Fig. 14 Time spent on different concurrent requests



(a) Throughput of PSC under different concurrent requests



(b) Throughput of RSC under different concurrent requests



(c) Throughput of ASC under different concurrent requests

Fig. 15 Throughput under different concurrent requests

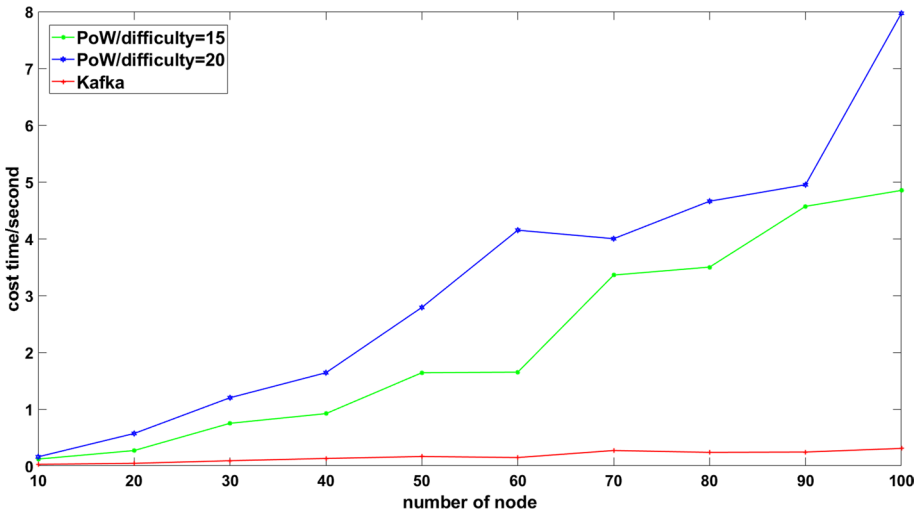


Fig. 16 Comparison of Kafka and PoW in the time of reaching consensus

resources. However, Kafka consensus can not only accomplish high throughput of transactions, but also provide sufficient fault tolerant workspace for consensus and ordering services. As shown in Fig. 16, in the second group of experiments, we compared the differences in consensus time between the Kafka consensus mechanism and the PoW consensus mechanism adopted in this scheme by setting the number of different nodes (between 10 and 100). The results show that this scheme can reach a consensus in a short time. Through the above two groups of experiments, it can be proved that this scheme can not only maintain high throughput in a large-scale request environment but also effectively reach consensus in a distributed system.

6 Conclusion

This paper combines blockchain technology with an attribute-based access control model to take full advantage of blockchain technology to break down information silos in medical data and safeguard the security and privacy of medical information. In addition, the interstellar file system is utilized in storage to ease the storage pressure on the blockchain. The scheme uses a distributed architecture to achieve dynamic fine-grained access. The deployment and invocation of the chain code are described in detail, and proof is given through experiments. In conclusion, this paper provides a practical reference for related research and can provide ideas for researchers. Future work could make improvements in the following areas.

1. This scheme is carried out on a single PC, and future consideration could be given to using clusters to further optimize the performance of the distributed system.
2. This scheme is based on the consensus mechanism of Kafka. To further reduce the arithmetic power and improve the consensus efficiency, a combination of other consensus algorithms can be considered in the future, such as using a consensus approach that combines Byzantine fault-tolerant algorithms with non-Byzantine fault-tolerant algorithms.

3. This paper combines IPFS and blockchain to alleviate the storage pressure of blockchain, but this is only a transitional stage, and in the future, we should consider solving the data storage problem from the blockchain.

Abbreviations

ABAC: Attribute-based access control; IPFS: Interplanetary file system; HIS: Hospital information system; EHR: Electronic health record; EMR: Electronic medical record; PKI: Public key infrastructure; P2P: Peer-to-Peer; SDK: Software development kit; ABACR: Attribute-based access control request; DHT: Distributed hash table; SFS: Scalable file service; HTTP: Hypertext transfer protocol; AS: Attributes of subject; AO: Attributes of object; AP: Attributes of permission; AE: Attributes of environment; Merkle DAG: Merkle directed acyclic graph; SHA256: Secure hash algorithm 256; CA: Certificate authority; Cert: Certificate file; Conf: Config file of the node; F(x)...: Functions defined in source code; CC: Chaincode in hyperledger fabric; ASC: Access smart contract; PSC: Policy smart contract; RSC: Record smart contract; Image: Docker image; TX: Transaction in blockchain; SDB: State database in hyperledger fabric; Cli: Blockchain system client; ABACP: Attribute-based access control policy.

Author contributions

ZS proposed and developed the new idea of the paper and drafted it. HD and LD have substantially revised it. WX and WZ conducted the data analysis and text combing. CC is responsible for supervision. All authors approved the submitted version. All authors read and approved the final manuscript.

Author information

Zhijie Sun received the BS degree in information management and information system from Henan Polytechnic University, and he is currently pursuing the MS degree in Shanghai Maritime University. His main research interests include blockchain technology and its applications and cryptography.

Dezhi Han received the BS degree from Hefei University of Technology, Hefei, China, the MS degree, and PhD 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 storage architecture, blockchain technology, cloud computing security, and cloud storage security technology.

Dun Li received the BS degree in Human Resource Management from the Huaqiao University, Quanzhou, China, in 2013 and the MS degree in Finance from the Macau University of Science and Technology, Macau, China, in 2015. He is currently doing his PhD 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.

Xiangsheng Wang is currently pursuing the PhD at Shanghai Maritime University. His main research interests include natural language processing, image processing, visual question answering, and machine learning.

Chin-Chen Chang received the PhD degree in computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982 and the BE and ME degrees in applied mathematics, computer and decision sciences from National Tsinghua University, Hsinchu, Taiwan, in 1977 and 1979, respectively. He was with National Chung Cheng University, Minxiong, Taiwan. Currently, he is a Chair Professor with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan, since 2005. His current research interests include database design, computer cryptography, image compression, and data structures. Prof. Chang was a recipient of many research awards and honorary positions by and in prestigious organizations both nationally and internationally, such as the Outstanding Talent in Information Sciences of Taiwan. He is currently a Fellow of the IEEE, a Fellow of the IEE, UK, and a Member of the IEICE.

Zhongdai Wu born in August 1976, is the Deputy General Manager, Chief Engineer, and Chief Information Officer of COSCO Shipping Technology Company Limited, with a doctoral degree, a researcher-level senior engineer, and a senior information manager of SASAC. He has more than 20 years of experience in shipping and logistics informatization construction. He has been responsible for the construction of various large-scale informatization projects of central enterprises and has presided over the completion of one Shanghai Key New Product, one Shanghai High-tech Achievement Transformation Project, one Shanghai Application Demonstration Project, and many software copyrights, and has rich experience in project planning and management. He has published more than ten academic papers, two of which were indexed by EI. He has been awarded as one of the top ten civilizational pacesetters of China Shipping Group, Shanghai New Long March Pioneer, Shanghai Federation of Trade Unions Scientific and Technological Innovation Talent, State-owned Assets Supervision, and Administration Commission Central Enterprise Knowledge-based Advanced Worker, Shanghai Young Post Leader, etc. Main research areas: shipping informationization, container management, ship and cargo management, logistics and supply chain technology research, cloud data center construction and management, network security situational awareness, shipping e-commerce, Internet of Things application, business intelligence technology, shipping big data application, ship satellite communication, etc.

Funding

This research is supported by the National Natural Science Foundation of China under Grant 61873160, Grant 61672338 and Natural Science Foundation of Shanghai under Grant 21ZR1426500.

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author details

[1]College of Information Engineering, Shanghai Maritime University, Shanghai, China. [2]Telecom SudParis, IMT, Institut Polytechnique de Paris, Palaiseau, France. [3]Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan. [4]The COSCO Shipping Technology Co., Shanghai, China.

Received: 30 December 2021 Accepted: 8 April 2022
Published online: 25 April 2022

References

1. V. Sima, I.G. Gheorghe, J. Subić, D. Nancu, Influences of the industry 4.0 revolution on the human capital development and consumer behavior: a systematic review. Sustainability **12**(10), 4035 (2020)
2. S. Schulz, R. Stegwee, C. Chronaki, Standards in healthcare data, in *Fundamentals of Clinical Data Science* (2019), pp. 19–36
3. X. Zhang, Y. Wang, Research on intelligent medical big data system based on hadoop and blockchain. EURASIP J. Wirel. Commun. Netw. **2021**, 1–21 (2021)
4. H. Li, D. Han, M. Tang, A privacy-preserving storage scheme for logistics data with assistance of blockchain. IEEE Internet Thing J. **9**, 4704–4720 (2021)
5. J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, N. Yu, Healthchain: a blockchain-based privacy preserving scheme for large-scale health data. IEEE Internet Things J. **6**(5), 8770–8781 (2019)
6. D. Han, N. Pan, K.-C. Li, A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection. IEEE Trans. Dependable Secure Comput. **19**, 316–327 (2022)
7. Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in *IEEE International Congress on Big Data (BigData Congress)* (2017), pp. 557–564
8. W. Fang, W. Chen, W. Zhang, J. Pei, W. Gao, G. Wang, Digital signature scheme for information non-repudiation in blockchain: a state of the art review. EURASIP J. Wirel. Commun. Netw. **2020**, 2020 (2020)
9. D. Li, D. Han, T.-H. Weng, Z. Zheng, H. Li, H. Liu, A. Castiglione, K.-C. Li, Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. Soft Comput. **26**, 4423–4440 (2021)
10. Q. Tian, D. Han, K.-C. Li, X. Liu, L. Duan, A. Castiglione, An intrusion detection approach based on improved deep belief network. Appl. Intell. **50**, 3162–3178 (2020)
11. B. Bhushan, A. Khamparia, K.M. Sagayam, S.K. Sharma, M.A. Ahad, N.C. Debnath, Blockchain for smart cities: A review of architectures, integration trends and future research directions. Sustain. Cities Soc. **61**, 102360 (2020)
12. P.K. Sharma, J.H. Park, Blockchain based hybrid network architecture for the smart city. Future Gener. Comput. Syst. **86**, 650–655 (2018)
13. H. Liu, D. Han, D. Li, Fabric-IoT: a blockchain-based access control system in IoT. IEEE Access **8**, 18207–18218 (2020)
14. D. Wang, H. Wang, Y. Fu, Blockchain-based IoT device identification and management in 5g smart grid. EURASIP J. Wirel. Commun. Netw. **2021**, 1–19 (2021)
15. D. Han, Y. Zhu, D. Li, W. Liang, A. Souri, K.-C. Li, A blockchain-based auditable access control system for private data in service-centric IoT environments. IEEE Trans. Ind. Inform. **18**, 3530–3540 (2021)
16. Y. Shan, Y. Mai, Research on sports fitness management based on blockchain and internet of things. EURASIP J. Wirel. Commun. Netw. **2020**, 2020 (2020)
17. D. Li, D. Han, N. Crespi, R. Minerva, Z. Sun, Fabric-SCF: a blockchain-based secure storage and access control scheme for supply chain finance. arXiv preprint. arXiv:2111.13538 (2021)
18. C. Wang, X. Cheng, J. Li, Y. He, K. Xiao, A survey: applications of blockchain in the internet of vehicles. EURASIP J. Wirel. Commun. Netw. **2021**, 1–16 (2021)
19. Z. Zhou, B. Wang, Y. Guo, Y. Zhang, Blockchain and computational intelligence inspired incentive-compatible demand response in internet of electric vehicles. IEEE Trans. Emerg. Top. Comput. Intell. **3**, 205–216 (2019)
20. M. Cui, D. Han, J. Wang, An efficient and safe road condition monitoring authentication scheme based on fog computing. EEE Internet Things J. **6**, 9076–9084 (2019)
21. M. Cui, D. Han, J. Wang, K.-C. Li, C.-C. Chang, ARFV: an efficient shared data auditing scheme supporting revocation for fog-assisted vehicular ad-hoc networks. IEEE Trans. Veh. Technol. **69**, 15815–15827 (2020)
22. H. Liu, D. Han, D. Li, Blockchain based trust management in vehicular networks, in *BlockSys* (2020)
23. K.A. Harthy, F.A. Shuhaimi, K.K.J.A. Ismaily, The upcoming blockchain adoption in higher-education: requirements and process, in *4th MEC International Conference on Big Data and Smart City (ICBDSC)* (2019), pp. 1–5
24. F.P. Oganda, N. Lutfiani, Q. Aini, U. Rahardja, A. Faturahman, Blockchain education smart courses of massive online open course using business model canvas, in *2nd International Conference on Cybernetics and Intelligent System (ICORIS)* (2020), pp. 1–6
25. D. Li, D. Han, Z. Zheng, T.-H. Weng, H. Li, H. Liu, A. Castiglione, K.-C. Li, MOOCsChain: a blockchain-based secure storage and sharing scheme for MOOCs learning. Comput. Stand. Interfaces **81**, 103597 (2021)
26. S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, F. Wang, Blockchain-enabled smart contracts: architecture, applications, and future trends. IEEE Trans. Syst. Man Cybern. Syst. **49**, 2266–2277 (2019)
27. D. Li, D. Han, H. Liu, Fabric-chain & chain: a blockchain-based electronic document system for supply chain finance, in *BlockSys* (2020)

28. J. Li, W. Yao, Y. Zhang, H. Qian, J. Han, Flexible and fine-grained attribute-based data storage in cloud computing. IEEE Trans. Serv. Comput. **10**, 785–796 (2017)
29. A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec: using blockchain for medical data access and permission management, in *2nd International Conference on Open and Big Data (OBD)* (2016), pp. 25–30
30. K. Fan, S. Wang, Y. Ren, H. Li, Y. Yang, Medblock: Efficient and secure medical data sharing via blockchain. J. Med. Syst. **42**, 1–11 (2018)
31. A.F. da Conceição, F.S.C. da Silva, V. Rocha, A. Locoro, J.M. Barguil, Electronic health records using blockchain technology. arXiv:1804.10078 (2018)
32. G. Yang, C. Li, A design of blockchain-based architecture for the security of electronic health record (EHR) systems, in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (2018), pp. 261–265
33. S. Kushch, S. Ranise, G. Sciarretta, Blockchain tree for ehealth, in *IEEE Global Conference on Internet of Things (GCIoT)* (IEEE, 2019), pp. 1–5
34. Q. Xia, E.B. Sifah, A. Smahi, S. Amofa, X. Zhang, BBDS: blockchain-based data sharing for electronic medical records in cloud environments. Information **8**, 44 (2017)
35. A. Zhang, X. Lin, Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. J. Med. Syst. **42**, 1–18 (2018)
36. S. Zhang, A. Kim, D. Liu, S.C. Nuckchadyy, L. Huangy, A. Masurkary, J. Zhangy, L.P. Karnatiz, L. Martínez, T. Hardjono, M. Kellis, Z. Zhang, Genie: a secure, transparent sharing and services platform for genetic and health data. arXiv:1811.01431 [cs.CR] (2018)
37. J. Liu, X. L. Li, L. Ye, H. Zhang, X. Du, M. Guizani, BPDS: a blockchain based privacy-preserving data sharing for electronic medical records, in *IEEE Global Communications Conference (GLOBECOM)* (2018), pp. 1–6
38. R. Qiao, X.-Y. Luo, S. Zhu, A.-D. Liu, X. Yan, Q. Xian Wang, Dynamic autonomous cross consortium chain mechanism in e-healthcare. IEEE J. Biomed. Health Inform. **24**, 2157–2168 (2020)
39. Y. Chen, S. Ding, Z. Xu, H. Zheng, S. Yang, Blockchain-based medical records secure storage and medical service framework. J. Med. Syst. **43**, 1–9 (2018)
40. B.S. Egala, A.K. Pradhan, V. Badarla, S.P. Mohanty, Fortified-chain: a blockchain-based framework for security and privacy-assured internet of medical things with effective access control. IEEE Internet Things J. **8**, 11717–11731 (2021)

Publisher's Note