On-Chain and Off-Chain Collaborative Management System Based on Consortium Blockchain

Kete Wang[1(✉)], Yong Yan[2], Shaoyong Guo[1], Xin Wei[1], and Sujie Shao[1]

[1] Beijing University of Posts and Telecommunications, Beijing 100000, China
wangkete@bupt.edu.cn

[2] State Grid Zhejiang Electric Power Co., Ltd., Electric Power Research Institute, Zhejiang 310000, China

Abstract. The blockchain system can provide a trust infrastructure for sharing data among untrusted parties. However, storing the original shared data directly on the blockchain is not suitable for large-scale data sharing scenarios. Therefore, we designed a data sharing system architecture in which data hashing and response records are stored on the blockchain and the original data is stored in the off-chain database. This architecture can alleviate the system overload and protect privacy problems to a certain extent. This paper proposes a three-tier system structure to ensure the function of the network. Subsequently, formulate request rules, deploy smart contracts, and build a platform based on the alliance chain. Finally, the system functions and performance are analyzed and compared through experiments. The results show that the system can realize efficient and transparent information sharing while satisfying on-chain and off-chain collaborative management, and the system has certain advantages in function, overall performance and throughput performance.

Keywords: Consortium blockchain · Collaborative management · Data share

1 Introduction

With the development of society, the use of identification is more and more frequent, and different types of identification are formed in different ways, but they all have security problems such as easy tampering and poor credibility. Compared with traditional physical storage evidence, review and certification is more complicated. When the identification is stored in a centralized manner, once the center is attacked or tampered with externally or internally, the credibility will decrease. In addition, to ensure the security of storage, the electronic storage of evidence often needs to use multiple backup methods, which will cause problems such as high storage costs. The identification has a strong relevance to the data, but it is difficult to support the data because of the difficulty of authentication, the large quantity, and the high cost of storage [1].

With the development of Internet technology, centralized architecture can no longer meet the requirements of security and performance. The researchers then turned to distributed storage and cloud computing, but the platform was vulnerable to DDos attacks and ignored issues such as authentication. The emergence of blockchain provides a feasible solution to the problems of traditional electronic storage of evidence [2]. Blockchain technology [3, 4] has a unique block-chain structure to store data, as well as timestamps, cryptography, consensus mechanisms, peer-to-peer communication, and distributed storage, which are jointly maintained by multiple parties to achieve decentralization and trusted data, hard-to-tamper target. Blockchain can build trust and centralization, and distributed storage protects electronic evidence. Electronic evidence includes transaction information and time stamp storage in summary form. Multiple parties jointly maintain consistency, reducing the possibility of tampering and making it more secure.In terms of data sharing, due to the lack of mutual trust between different companies or different government departments, the risks of data leakage and improper use, and the differences in administrative interests between companies or government departments, many data owners are unwilling to share. In terms of data privacy, literature [5] et al. proposed an EHR sharing protocol based on the security and privacy protection of the blockchain by using the decentralization, anonymity, unforgeability and verifiability of the blockchain. In the solution, the data requester can search for the required keywords from the data provider, find the relevant HER on the blockchain, and obtain the re-encrypted ciphertext from the cloud server after obtaining the authorization of the data owner. This solution mainly Use searchable encryption and conditional proxy re-encryption to achieve data security, privacy protection and access control.

In terms of data access control and security, literature [6] according to most database systems and enterprise information systems are role-based access control technology, but due to the simple role access control, its flexibility and control granularity sometimes cannot meet the actual access control. Therefore, a security access control model based on RBACV1 and ABAC is proposed to solve this problem.

In terms of credible deposits, literature [7] proposes an Ethereum trusted deposit framework based on smart contracts for the data management problems of the Ethereum platform, and then through centralized data unified processing, certified data distributed storage and efficient dynamics Forensic mechanism to achieve. Finally, the system development scheme design based on smart contract shows the feasibility of the mechanism. Hou Yibin et al. [8] tried and studied the combination of blockchain technology and electronic evidence technology to highlight the digitalization of electronic evidence and the security and reliability of blockchain technology. The electronic evidence storage system architecture in the form of batch packaging of evidence improves the efficiency of evidence storage.

Therefore, the main technical contributions of this paper are summarized as follows:

- This article proposes a fabric-based on-chain and off-chain data collaborative management mechanism based on the problems of easy tampering, low data trust, unguaranteed security, data islands, and large storage capacity in traditional storage methods.
- Through the designed on-chain smart contract for certification and virtualized resource pool as an off-chain database, this article uses blockchain technology to make the certification data safe and reliable, and at the same time alleviate the storage pressure on the chain. Consortium chain nodes are jointly maintained by multiple institutions,

and each consortium block chain node (CBN) is executed by a private server belonging to a trusted authority.
- The system establishes a two-way communication between organization A and organization B. After data collection or addition, the identification is stored in the blockchain through a smart contract, and the data is added to the shared database. The blockchain will serve as a depository. Including the hash of the verification data and the organization to which the recorded data belongs.

2 Related Work

2.1 Hyperledger Fabric

HyperLedger Fabric [9, 10] is a modularized distributed ledger solution platform and the underlying basic framework of a permission blockchain. It has the advantages of convenient expansion and pluggability, and is suitable for enterprise-level applications. In the ledger, the data blocks are linked in sequence in the order of generation time, and cryptography [11], consensus algorithm [12] and other methods are used to ensure the uniformity, non-tampering and unforgeability of the data of the ledger. Compared with other public chains, HyperLedger Fabric's differences are mainly reflected in the two aspects of privateness and permission. Members of its organization can register through membership services to ensure the security of platform access. The main structure of Hyperledger is shown in Fig. 1.

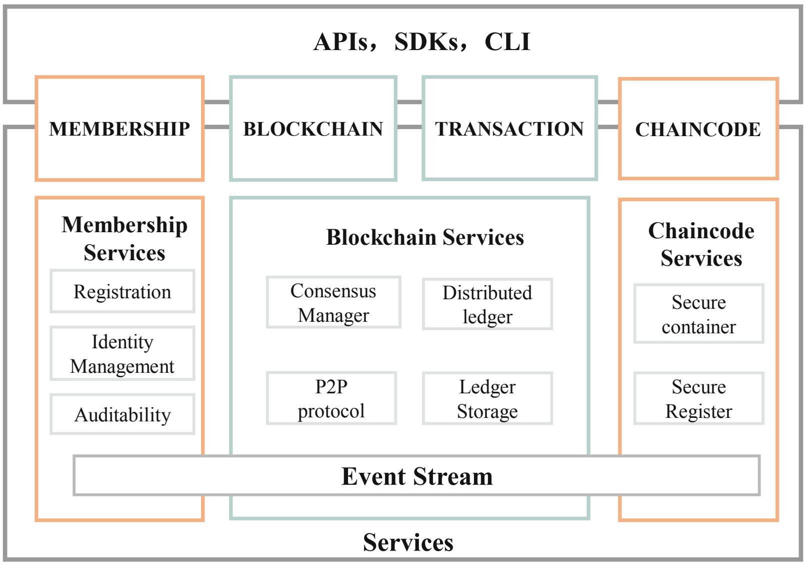


Fig. 1. Hyperledger architecture

Member management services ensure the security of Fabric platform access and provide system members with registration, management and audit functions. The blockchain service is the core part, which provides support for the main functions of the blockchain, including consensus mechanism management, implementation of distributed ledgers, storage of ledgers, and communication between nodes. The chain code service part provides an environment for the deployment and operation of smart contracts.

2.2 Transaction Process

In the Fabric network environment, its nodes can be divided into Endorsing peer, Committing peer, Orderer peer, Anchor peer and Leading peer according to different functions. Among them, the Endorsing peer will endorse the transaction according to the called smart contract and return it to the client. Committing peer is responsible for verifying transaction data and saving it in the ledger. The Orderer node is responsible for sorting transactions and creating blocks. Anchor peer is responsible for cross-organization communication. Leading peer is the representative of all members in the organization, responsible for connecting to the Orderer node and broadcasting the received messages. The specific transaction process is described in Fig. 2 below.

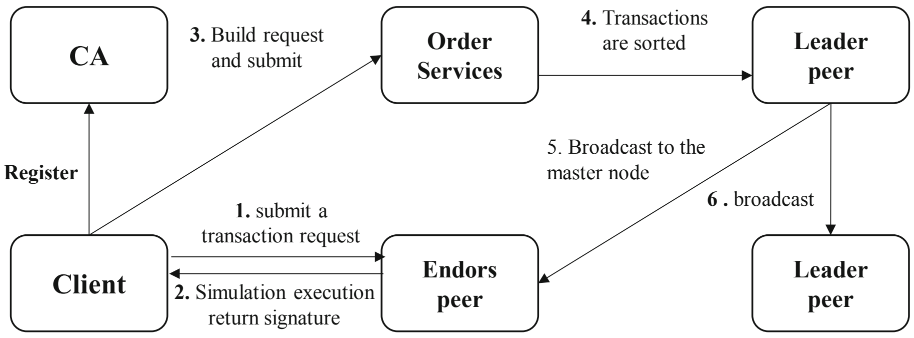

CA
3. Build request and submit
Order Services
4. Transactions are sorted
Leader peer
Register
5. Broadcast to the master node
6 . broadcast
1. submit a transaction request
Client
2. Simulation execution return signature
Endors peer
Leader peer

Fig. 2. Description of the transaction process

3 Requirements Analysis

3.1 Application Scenario

It shows an application scenario of on-chain storage of certificates and off-chain data transmission. In this scenario, there are 4 participants A, B, C, and D. Each participant has its own data. The original data is in ciphertext. The form is stored on the cloud server. The identification of the original data (that is, the data catalog information, including the basic description, category, owner, etc. of the data) and its hash digest are stored on the blockchain. Suppose that due to certain services, node D needs to obtain data set R, and the client of node D finds that node B has data set R by querying the catalog information on the chain. Therefore, the client of the D node initiates a data acquisition request to the B node, and the B node uses the public key of the D node client to symmetrically

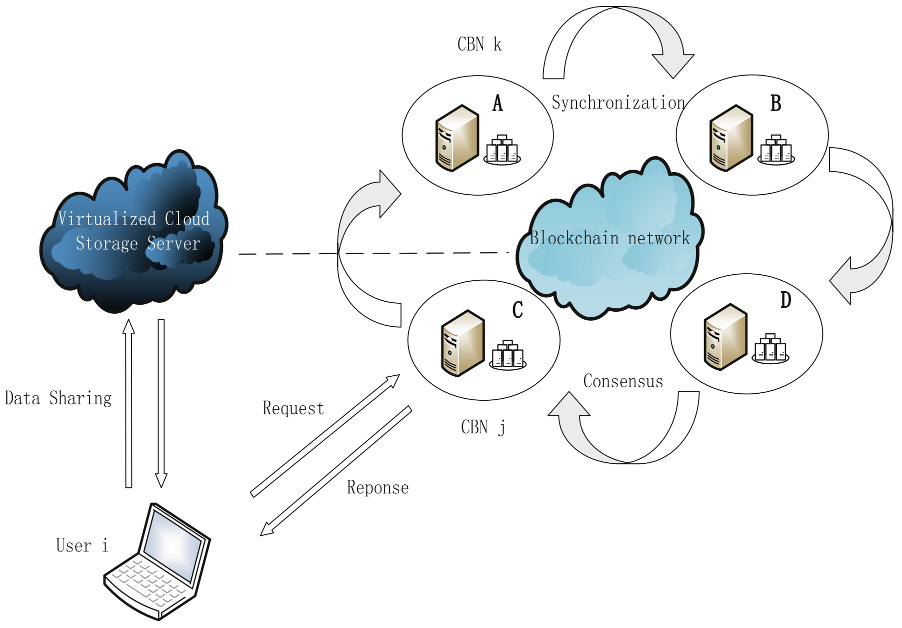


Fig. 3. Application scenario

encrypt the key K and the vector IV to the D node. Finally, the D client downloads the data to the cloud server according to the received information and verifies its hash, and uses the data set R legally after verification. In order to protect the respective rights of data owners and data users, the data request from the D node client and the B node's response are recorded on the blockchain, of which 4 nodes A, B, C, and D are all this data Witnesses of requests and responses. As shown in Fig. 3.

3.2 Functional Module

The user registration module mainly means that the user must submit a request and perform authentication registration first to participate in the system, and only after passing the system audit can the user participate in the system, which ensures that the identity of the system participant is clear. The system will also record the user's operating behavior to ensure that the responsibility of the electronic deposit data can be traced.

The original data storage module is mainly responsible for passing the user's original data to the logic layer through the front end, and to the cloud storage server through the call interface. After obtaining the hash value of the file, it is encrypted by the access control module for the next step of identification On the chain. In the same way, the original data storage module needs to call the interface through access permission control to download the original data from the cloud server.

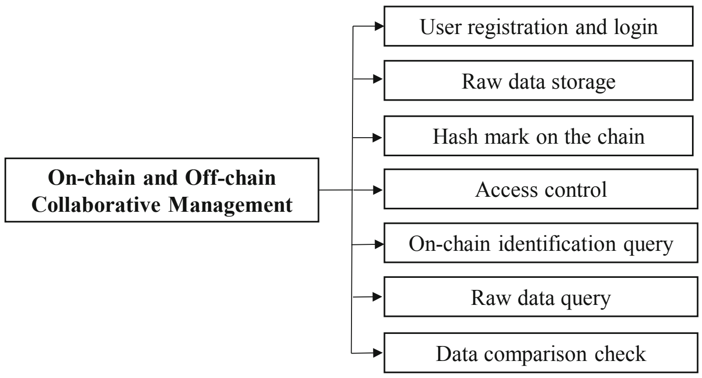

On-chain and Off-chain Collaborative Management
User registration and login
Raw data storage
Hash mark on the chain
Access control
On-chain identification query
Raw data query
Data comparison check

Fig. 4. System module

The hash mark on-chain module is mainly responsible for storing the summary of the user's data information in the ledger. The user submits an application to the system before the data is uploaded, and after the application is approved, the unique identification ID assigned by the system will be obtained to identify user information. The chain can handle different service requirements and control the participation of nodes through different channels. In the process of chaining, smart contracts approved by judicial review are used for chaining to avoid uncertain factors caused by human intervention. In the same way, the identification query module on the chain uses the unique identification ID to request the corresponding node to perform the identification query on the blockchain.

The main purpose of the access control module is to enhance the privacy of data uploaded by users in the system. When user A does not want other users in the same channel to query the hash value of the electronic evidence file uploaded by him, and then query the file he uploaded, he can use symmetric encryption for the hash value of the file before the data is uploaded to the chain encrypt it in the method, and then use the public key of user B who is authorized to access the encrypted key and initial vector to asymmetrically encrypt, and then send the encrypted data to user B who is authorized to access through the service layer, and the user who is authorized to access B can use its private key to decrypt the encryption key and initial vector of user A, and then decrypt the hash value of the file.

The data comparison verification module mainly verifies the correctness and traceability of data or files. Check the hash results of the blockchain and the cloud database, and return the two results to the client if they are correct. As shown in Fig. 4.

4 Design and Implementation

4.1 Architecture Design

This paper establishes a system model that can realize encrypted data transmission, identity verification and secure data storage. The entire system architecture is divided into three layers, namely the user layer, the blockchain layer and the data storage layer. As shown in Fig. 5.

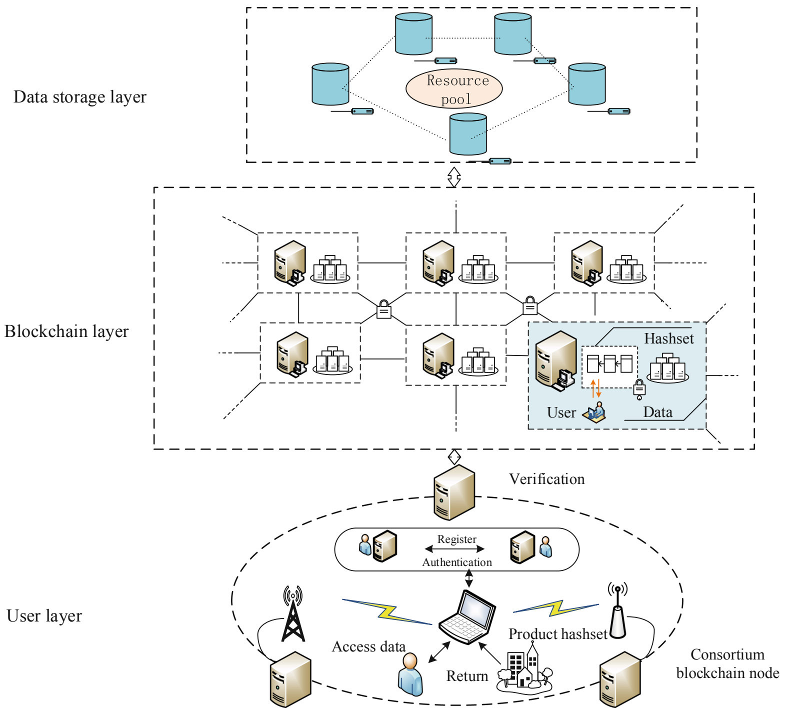


Fig. 5. System architecture

A. ***User layer***
The user layer, including the process of data collection, encryption, and transmission, interacts with the storage layer and the blockchain layer through visual interfaces such as the web. In addition, including user registration and authentication. This layer implements data storage, query, and credibility verification, provides access interfaces, and automatically executes after triggering smart contracts. For example, trusted verification, by querying database data and blockchain authentication information, the hash is compared and displayed on the web. The user interacts with the system through the interface.

B. ***Blockchain layer***
The node maintains the consistency of the Consortium blockchain according to the preset consensus mechanism, and verifies the integrity of the data and the identity of the corresponding user by checking identity information through signatures and certificates. The blockchain layer uses existing access points to access the CBN server for triggering. After identity authentication, the data is packaged and hashed into the blockchain through the smart contract, and the original data is packaged and stored in the database. In other words, the node packs the transactions generated within a specific time window into the data block of the Merkle tree structure. In addition, the client accesses the blockchain through the smart contract, parses the required

data through the smart contract, performs privacy protection processing on the original data owner, and performs credibility verification through hash comparison to ensure that the data is true and credible and returned to the requester. In short, due to the consensus-based distributed data verification mechanism, the system brings an immutable, anonymous, irrevocable and traceable blockchain distributed ledger for the auditable data in the system verification.

C. ***Data storage layer***
Since the blockchain is not designed for large-scale database storage, a collaborative management mechanism for verification on the off-chain storage chain should be required. Instead, only data identifiers such as identity verification information, hash values, and data signs are stored on the alliance chain nodes. Store a large amount of user data in a virtualized resource pool.

4.2 Smart Contract Design

Smart contracts in the Fabric network are also called chain codes. They run in Docker containers and are mainly written in Golang. All peers in the system can call the contract to access data hash transaction information after adding the chain code. The core smart contract part of this system is mainly for hash mark on-chain and on-chain query. The created transaction includes user number, transaction number, timestamp, hash, type, description, and attribution. The structure is expressed as Hd = (User, ID, Timestamp, Hash, Type, Describe, Belong). When operating, use the system package provided by Fabric to communicate with the blockchain network, namely Shim package and Peer package. The Shim package contains the interface method for the interaction between the smart contract and the Hyperledger, which provides the context of the Hyperledger network for the operation of the chaincode.

Table 1. Hash related attributes

Method	Request	Input	Output	Description
Init	GET	N/A	Boolean	Initialize the chaincode and return a boolean
Invoke	GET	N/A	Boolean	Forward parameters to the corresponding method
Regist	POST	HASH, ID	TxID	Register hash, mark return transaction ID
SetHash	GET	ID	HASH	Set hash mark
QueryHash	GET	ID	Data	Query hash mark
Indentify	GET	Data, HASH	Boolean	Verify the credibility and correctness of data

To call the chain code in Fabric to query the ledger information, the system must implement the ChaincodeStubInterface interface under the shim package in the chaincode chainCode. The chain code provides a hash service for users and mainly defines the related functions of Table 1. According to different request types of chaincode calling methods, transactions are divided into query and invoke. For example, simple query of ledger information will directly send query; if it involves update and increase, etc., the invoke transaction will be sent, waiting for other nodes to endorse to complete the transaction (Table 2).

Table 2. Chaincode related functions

Property	Type	Description
User	String	User ID
ID	Int	Transaction ID
Timestamp	Date	Time to record data
Hash	String	File or data hash
Type	String	Data type
Describe	String	Data description
Belong	String	Data Ownership Organization

The hash on the chain is mainly stored in the ledger through the PutState method in the shim package in Fabric. First, you need to define a suitable JSON data structure to store the data that needs to be on the chain. Get the parameters through the ChaincodeStubInterface in Shim, and then you need to check Whether the format and content of the upload parameters meet the requirements, in addition, the GetState method needs to be used to verify whether the data already exists in the ledger. When the data is verified to meet the requirements, it is converted into a JSON string and stored and the PutState method is called to store the data on the chain. If successful, the result of the chain is returned. The specific algorithm flow is summarized as follows (Table 3).

Table 3. Chaining and storage

```
Algorithm 1
Input：  Hd（User,  ID,  Timestamp,  Hash,  Type,  Describe,  Belong）
Output：(Putstate Result,  Event,  TxID)
1:          Get parameters Hd through ChaincodeStubInterface
2:          If the number of Hd is not 7, then return an error message
3:          Check whether the number ID already exists by GetState(ID)
4:          If number exist ,then return error
5:          If the number does not exist, convert the information in Hd into a JSON string
6:          Through Putstate function of shim to storage into hyperledger
7:          Return Result
```

Table 4. Query and verification

```
Algorithm 2
Input:   T(ID)
Output:  (Getstate result, Query result)
1:    Get parameters Hd through ChaincodeStubInterface
2:    If the number of Hd is not 1, then return an error message
3:    Read the data content of Number ID through GetState(ID)
4:    Determine whether the read data is empty, if so, return an error message
5:    Return the query result.
```

The smart contracts queried on the chain are mainly used to obtain data from the ledger through the GetState method of the shim package in Fabric. The user can use the ID when hashed on the chain as the query condition, first check whether the input parameters meet the requirements, and then query the data information numbered ID according to the GetState method, and judge whether the retrieved data is null or whether there is an error. Finally, the query result is returned. The specific algorithm flow is summarized as follows (Table 4).

Table 5. Permission access control

```
Algorithm 3
Input:   (Hash, K, IV, $PK_B$)
Output:  (E_Hash, E_K, E_IV)
1:    The hash value of the input data
2:    XOR hash to initial vector, temp $\leftarrow$ hash to IV XOR
3:    The results of step 2 are encrypted symmetrically_ Hash $\leftarrow$ AES_ E (K , Temp)
4:    Using $PK_B$ to encrypt K and IV asymmetrically, E_ K, E_ IV $\leftarrow$ ECC_ E ($PK_B$, K, IV)
5:    Returns the encrypted hash E_ Hash and ciphertext e_ K and ciphertext e_ IV
```

The design scheme of system privacy protection mainly realizes data privacy protection by encrypting the data submission link. When user A does not want the hash value of the uploaded file to be seen by all users in the channel, he can use the AES encryption algorithm to encrypt the hash value of the file before uploading the data to the chain. The process is described as follows: User A uses the key K and the initial vector IV to encrypt the data. The encryption function is defined as AES_E(K, Hash), where K is the key and Hash is the hash of the data returned by the database sql. When user B is expected to view the data, user B's public key PK_B can be used to perform ECC asymmetric encryption on the key K and the initial vector IV. The encryption function is defined as ECC_E(PK_B K, IV), and the encrypted text is ciphertext E_K and ciphertext E_IV are sent to user B through the business layer after encryption (Table 5).

5 Evaluation

The first part introduces the required hardware configuration and basic software environment. The second part introduces the construction of Hyperledger fabric and node introduction. The third part introduces the function of data storage query on the system chain and the system access control. The fourth part tests and compares the system throughput.

5.1 Environment Configuration

This article installs a virtual machine in the host and deploys the Hyperledger to run in the virtual machine. The required software environment is shown in Table 6 below.

Table 6. Software environment

Software environment	Detailed information
OS	Ubuntu 20.04.1 LTS
Docker	v19.03.12
Docker-compose	v1.26.0
Go	go1.13.8
Hyperledger Fabric	v1.2.0

5.2 Operating Environment

Set up four different types of nodes in the fabric operating environment. As shown in the following Table 7.

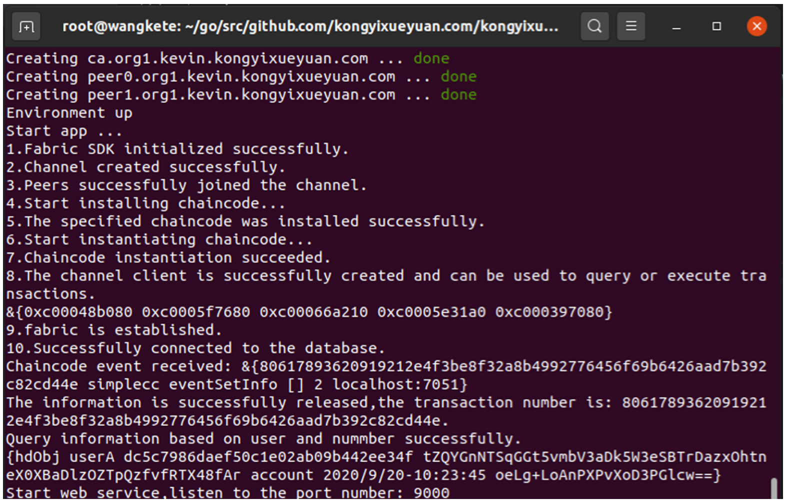
The setup process of Hyperledger's operating environment is as follows:

1. Generate the peer node and orderer node to generate the certificate and key. In this article, two peer organizations are set up, each organization contains two nodes, and an orderer node is set.
2. Use the encryption tool configtxgen to read the configuration information in the configtx.yaml file:

Table 7. Fabric node types

Node name	Node type
Leveldb	Database node
Peer	Bookkeeping node
Orderer	Sort node
CA	CA node

3. Create a channel and read information. Finally, create a container according to the docker compose startup image, add each node to the created channel, and build it. After the network operating environment of Hyperledger is built, the chain code needs to be installed and instantiated before the chain code can be used normally. The chain code installation process is shown in the Fig. 6.


root@wangkete: ~/go/src/github.com/kongyixueyuan.com/kongyixu...

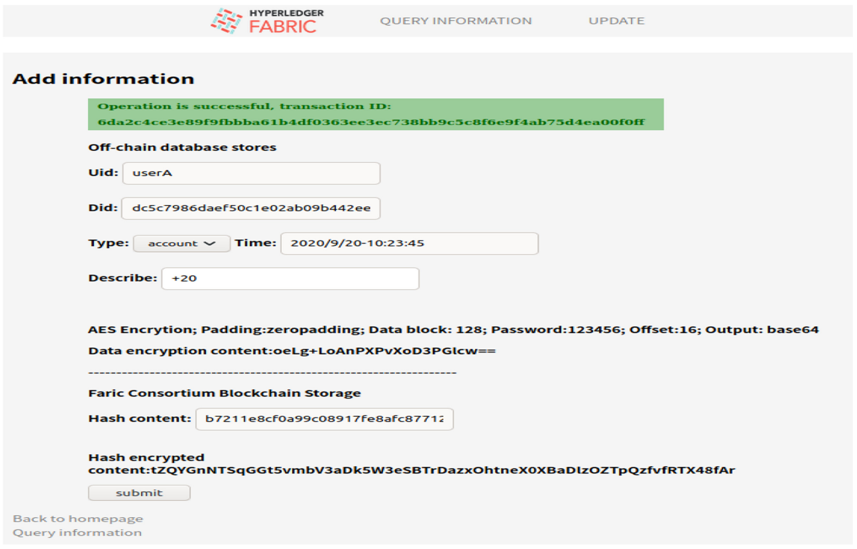
```
Creating ca.org1.kevin.kongyixueyuan.com ... done
Creating peer0.org1.kevin.kongyixueyuan.com ... done
Creating peer1.org1.kevin.kongyixueyuan.com ... done
Environment up
Start app ...
1.Fabric SDK initialized successfully.
2.Channel created successfully.
3.Peers successfully joined the channel.
4.Start installing chaincode...
5.The specified chaincode was installed successfully.
6.Start instantiating chaincode...
7.Chaincode instantiation succeeded.
8.The channel client is successfully created and can be used to query or execute tra
nsactions.
&{0xc00048b080 0xc0005f7680 0xc00066a210 0xc0005e31a0 0xc000397080}
9.fabric is established.
10.Successfully connected to the database.
Chaincode event received: &{80617893620919212e4f3be8f32a8b4992776456f69b6426aad7b392
c82cd44e simplecc eventSetInfo [] 2 localhost:7051}
The information is successfully released,the transaction number is: 8061789362091921
2e4f3be8f32a8b4992776456f69b6426aad7b392c82cd44e.
Query information based on user and nummber successfully.
{hdObj userA dc5c7986daef50c1e02ab09b442ee34f tZQYGnNTSqGGt5vmbV3aDk5W3eSBTrDazxOhtn
eX0XBaDlzOZTpQzfvfRTX48fAr account 2020/9/20-10:23:45 oeLg+LoAnPXPvXoD3PGlcw==}
Start web service,listen to the port number: 9000
```

Fig. 6. Fabric successfully built

5.3 System Functions

To verify the add function of the system, the user uploads the file through the database provided by the web to get the hash value. After the members of the organization encrypt the hash, the ID, user identity, and timestamp assigned by the system are uploaded to the blockchain through the SDK. When UserA adds data to the system, the page is shown in the Fig. 7, and the terminal is Fig. 9.

To verify the query function of the system, user UserA initiates a request to UserD. UserD uses A's public key to encrypt the password and sends it to A. After a series of



Add information

Operation is successful, transaction ID: 6da2c4ce3e89f9fbbba61b4df0363ee3ec738bb9c5c8f6e9f4ab75d4ea00f0ff

Off-chain database stores

Uid: userA

Did: dc5c7986daef50c1e02ab09b442ee

Type: account Time: 2020/9/20-10:23:45

Describe: +20

AES Encrytion; Padding:zeropadding; Data block: 128; Password:123456; Offset:16; Output: base64

Data encryption content:oeLg+LoAnPXPvXoD3PGlcw==

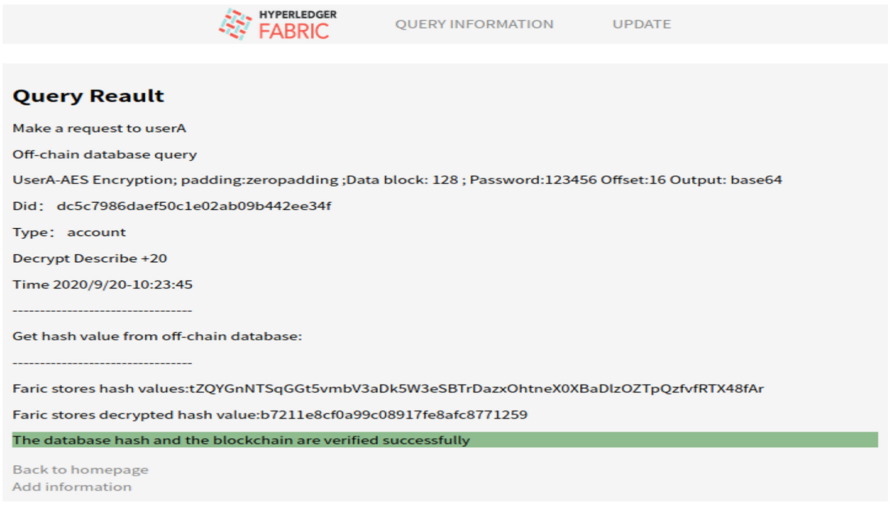
Faric Consortium Blockchain Storage

Hash content: b7211e8cf0a99c08917fe8afc8771

Hash encrypted content:tZQYGnNTSqGGt5vmbV3aDk5W3eSBTrDazxOhtneX0XBaDlzOZTpQzfvfRTX48fAr

Back to homepage
Query information

Fig. 7. Data Insertion page



Query Reault

Make a request to userA

Off-chain database query

UserA-AES Encryption; padding:zeropadding ;Data block: 128 ; Password:123456 Offset:16 Output: base64

Did: dc5c7986daef50c1e02ab09b442ee34f

Type: account

Decrypt Describe +20

Time 2020/9/20-10:23:45

Get hash value from off-chain database:

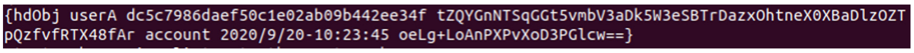
Faric stores hash values:tZQYGnNTSqGGt5vmbV3aDk5W3eSBTrDazxOhtneX0XBaDlzOZTpQzfvfRTX48fAr

Faric stores decrypted hash value:b7211e8cf0a99c08917fe8afc8771259

The database hash and the blockchain are verified successfully

Back to homepage
Add information

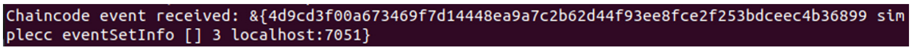
Fig. 8. Query and verify records page



```
{hdObj userA dc5c7986daef50c1e02ab09b442ee34f tZQYGnNTSqGGt5vmbV3aDk5W3eSBTrDazxOhtneX0XBaDlzOZT
pQzfvfRTX48fAr account 2020/9/20-10:23:45 oeLg+LoAnPXPvXoD3PGlcw==}
```

Fig. 9. Inserted terminal

decryptions, A obtains the hash, and then requests the database to get the data decrypted through the web and verify it. After the verification is successful, it is displayed on the



```
Chaincode event received: &{4d9cd3f00a673469f7d14448ea9a7c2b62d44f93ee8fce2f253bdceec4b36899 sim
plecc eventSetInfo [] 3 localhost:7051}
```

Fig. 10. Query and verify terminal

screen, the query result is shown in Fig. 8, and the query background record is shown in Fig. 10.

5.4 System Performance

This performance test mainly uses the Caliper tool for testing, which is a blockchain performance benchmark framework and allows users to use predefined use cases to test different blockchain design solutions and obtain performance test results. The system tested the throughput of different numbers of nodes, and the throughput and latency of different read and write times.

When the number of nodes is an experimental variable, the nodes grow from 0 to 35 with a step size of 5. Repeat the experiment under different nodes to take the average value, and the experimental results are shown in Fig. 11.

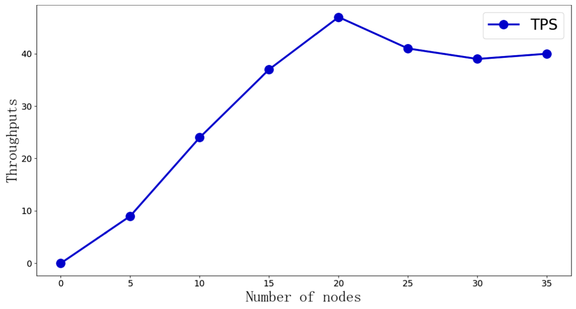


Fig. 11. The effect of nodes on throughput

The throughput of the write function reaches its peak at 20 requests, decreases slightly with the increase in the number of accesses, then increases slightly, and finally stabilizes. The throughput of the read function shows an upward trend as the number of requests increases, indicating that the system can withstand a certain scale of access requests.

By comparison, in the Hyperledger fabric, the system reads without submitting new transactions, and does not interact with the ordering node and the submitting node. Therefore, the system read throughput is better than the write throughput. The specific data is shown in Fig. 12.

The delay of the write function is basically 0.3 or less, reaching a peak at 100 times. As the number of accesses increases, the average delay decreases and tends to stabilize. The delay of reading the function is basically lower than 0.1, and the average delay decreases with the increase of the number of visits, reaching the lowest at 1000 times. The specific data is shown in Fig. 13.

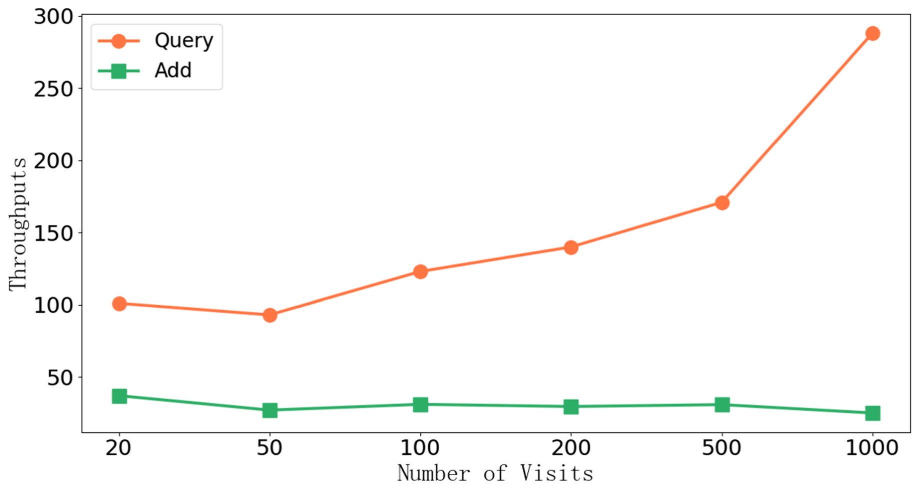


Fig. 12. Throughput test

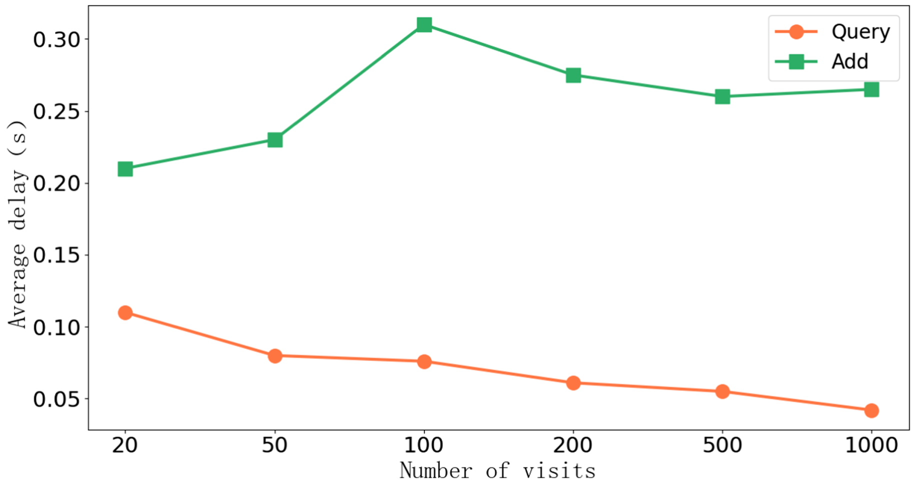


Fig. 13. Latency test

6 Conclusion

The rapid development of blockchain technology and the high level of social attention have made blockchain applications in various fields. This paper proposes a fabric-based on-chain and off-chain data collaborative management mechanism based on the problems of easy tampering, low data trust, unguaranteed security, and large storage capacity in traditional storage methods. Through the designed on-chain smart contract for certification and virtualized resource pool as an off-chain database, this article uses blockchain technology to make the certification data safe and reliable, and at the same time alleviate the storage pressure on the chain. The nodes of the alliance chain are jointly maintained by multiple institutions, etc., which solves the problems of information opacity, sharing, and security to a certain extent. The system establishes channels in different organizations, nodes can access each other, and the encryption algorithm is used to solve the privacy problem, but the security problems and throughput problems in complex environments need to be further improved in follow-up research.

Acknowledgement. This work has supported by State Grid Corporation of China science and technology projected "Key Technologies Research and Application of High Elastic Power Grid based on blockchain" (5700-202019374A-0–0-00).

References

1. Shangang, Z., Chao, W.: Review and judgment on the evidence ability of electronic data. People's Procur. Semimon. **8**, 34–36 (2018)
2. Xin, Z., Tao, L.: Research on the judicial deposit system of blockchain. Cyberspace Secur. **10**(7), 44–47+72 (2019)
3. Junfei, H., Jie, L.: Survey on blockchain Research. J. Beijing Univ. Posts Telecommun. **41**(2), 5–12 (2018)
4. Yong, Y., Feiyue, W.: Blockchain: the state of the art and future trends. Acta Automatica Sinica **42**(4), 481–494 (2016)
5. Wang, Y., Zhang, A., Zhang, P., et al.: Cloud-assisted EHR sharing with security and privacy preservation viaconsortium blockchain. IEEE Access **7**, 136704–136719 (2019)
6. Ding, X., Yang, J.: An access control model and its application in blockchain. In: 2019 International Conference (2019)
7. Didi, C., Wei, C.: Mechanism of trusted storage in Ethereum based on smart contract. J. Comput. Appl. **39**(4), 145–152
8. Hou, Y., Liang, X., Zhan, X.: Block chain based architecture model of electronic evidence system. Comput. Sci. **45**(S1), 361–364 (2018)
9. Cachin, C.: Architecture of the hyperledger blockchainfabric [EB/OL]. http://bytacoin.io/main/Hyperledger.pdf. Accessed 12 July 2016
10. Stallings, W.: Cryptography and network security: principles and practice. Int. Annals Criminol. **46**(4), 121–136 (1999)
11. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: practical decentralized coin mixing for bitcoin. In: Kuty łowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 345–364. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_20
12. Androulaki, E., Barger, A., Bortnikov, V., et al.: Hyperledgerfabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. Association for Computing Machinery, New York, NY, United States