Complex & Intelligent Systems (2023) 9:2865–2877
https://doi.org/10.1007/s40747-021-00617-1

ORIGINAL ARTICLE

TVS: a trusted verification scheme for office documents based on blockchain

Xue Zhai[1] · Shanchen Pang[1] · Min Wang[1] · Sibo Qiao[1] · Zhihan Lv[2]

Received: 15 June 2021 / Accepted: 3 December 2021 / Published online: 5 January 2022

Abstract

To realize the encryption of document information, authority authentication, and traceability of historical records, we propose a trusted verification scheme (TVS) for office documents to ensure security. Specifically, the scheme is realized by timestamps, smart contracts (or chaincode), and other blockchain technologies. It is based on the features of blockchain, such as security, credibility, immutability, and traceability of network behavior. And the TVS stores users and documents information through blockchain; it can monitor the state changes of office documents in real time by setting the trigger conditions of smart contracts. The experiment indicates that we have realized the real-time monitoring of data and the traceability of historical records. Moreover, we have achieved the purpose of document encryption and authority authentication, ensuring the authenticity and objectivity of data, avoiding the illegal tampering of malicious users to realize the trusted verification for documents.

Keywords Document security · Authority authentication · Blockchain · Smart contract

Introduction

Office documents, realizing information sharing and paperless office, are used in all walks of life; it not only improves the accuracy in the office but also raises the working efficiency. However, hacker attacks, network intrusion, and Trojan horse will bring harm to document transmission. They can cause vital documents lost and being tampered, or even the entire transfer system breakdown, so ensuring document security is crucial to users[1].

Blockchain provides a shared, tamper-resistant, and transparent transaction record, supporting the construction of applications that include trust, responsibility, and transparency. However, blockchain is mainly used for verifiable public transactions; it provides no privacy for individuals. In the existing works, the blockchain is used to develop a safe and reliable data sharing system to ensure the privacy, integrity, and fine-grained access control of the shared data. Mohammad et al. [2] established a data notarization system based on the blockchain, verifying the authenticity of shared documents in real time. Yuan et al. [3] proposed a new scheme that combines blockchain and CP-ABE; it stored data changes on the blockchain and implemented different access rights through ABE, realizing effective data supervision and privacy protection. Zhu et al. [4] used the decentralized feature of the blockchain, identity-based cryptography, and electronic signature to achieve electronic contract signing and certification of important documents. Wang et al. [5] proposed a secure and high-performance multi-party computing model, which combined the characteristics of the blockchain with security multi-party computation. It used proxy re-encryption for data sharing and an improved consensus algorithm for ensuring consistency between nodes, so that users could control data independently. Tian et al. [6] proposed a secure and efficient public auditing scheme for user operation behavior logs based on blockchain. It can remotely verify the integrity of log data in the cloud, resist collusion attacks between malicious users, and the cloud ser-

✉ Shanchen Pang
pangsc@upc.edu.cn

Xue Zhai
2417792534@qq.com

Min Wang
minwang2121@163.com

Sibo Qiao
siboqiao@126.com

Zhihan Lv
lvzhihan@gmail.com

[1] China University of Petroleum, Qingdao, Shandong, China

[2] The School of Computer Science, Qingdao University, Qingdao, Shandong, China

vice provider, and prevent cloud data from being damaged and tampered. Datta et al. [7] combined an edge computing paradigm with a blockchain setting and proposed a detailed three-layer edge computing architecture to improve the security of Secured Data Delivery for Forest Fire Surveillance. In addition, it proposed an energy-efficient prospective Leader selection algorithm and an optimal and dynamic drone trajectory algorithm to minimize energy consumption.

Blockchain is also used in the fields of document certification, asset trading, medical information protection [8–11], and IoT interaction. It not only solves the security loopholes in traditional technical solutions and optimizes the business models, but also improves transaction efficiency and ensures security [12]. To ensure the integrity of medical records and prevent it from being tampered, Brihat et al. [13] proposed an innovative method based on Merkle Tree to protect the integrity of medical records, it avoided mining to simplify operations, and replaced traditional audit trails with encrypted and secure counterparts, which improved the robustness of the medical records storage method. Jia et al. [14] proposed an identity-based cross-domain authentication scheme for IoT, using blockchain as a decentralized trust anchor, and identity-based self-authentication algorithms instead of PKI to achieve decentralized authentication; it ensured the autonomy and initiative of the security domain. Krishnan et al. [15] improve verifiable and immutable repositories by employing blockchain to design authentication in the IoT onboarding process. And they combined with blockchain and SxC security contracts, MUD-based behavioral fingerprinting, and Software-Defined-Networking (SDN), to design an integrated framework for managing the security of IIoT ecosystems. It can effectively ensure the proper functioning and performance of Industrial grade IoT devices (IIoT) in Industry 4.0 networks.

To prevent office documents from being tampered with by illegal users, monitoring the state changes of office documents in real time, and to achieve the purpose of traceability of historical records, we establish a trusted verification scheme for office documents based on blockchain. Specifically, we intend to use the *Hyperledger Fabric* as the experimental blockchain platform, node users, digitize users, and documents information, and divide users into groups by category. Then, the smart contract is designed according to the consensus rules reached between users. Through the uploading and calling of the smart contract, we realize the authority authentication, sharing, and historical records' tracing of documents.

The main contributions of this manuscript are summarized as follows:

1. This manuscript proposes a trusted verification scheme for office documents based on blockchain. In this scheme, we node users, digitize users, and documents information, and divide users into groups by category. In addition, we have designed a channel to provide privacy protection for members. It realizes the mutual cooperation and supervision of peer nodes in the blockchain network, and ensures the user's identity authentication and authority control.
2. The algorithms for initializing, querying, and modifying office document information using blockchain are proposed, and we implement it using smart contracts. Since the operation of the smart contract is not affected by external factors, it can obtain objective and accurate results, and realize the real-time storage of office documents and historical records tracing.
3. We use the *Hyperledger Fabric* as an experimental platform to verify the proposed scheme and algorithms. First, we successfully built the experimental environment to provide sufficient conditions for algorithm verification. Second, we deploy the chaincode in the blockchain environment and initialize the office documents. Finally, we realize all the functions of the proposed scheme and achieve the purpose of monitoring the state change of office documents in real time.

The rest of this manuscript is organized as follows. Section Related work reviews the related works. Section Overall model introduces the proposed overall model. Section Function modules gives the main function modules in detail. The experiment testing and evaluation are given in Sect.Experiment and evaluation. Section Conclusion and future work concludes this manuscript.

Related work

While realizing information sharing, sensitive information of office documents may lead to information leakage, the existing studies adopted identity authentication and access control technique to guarantee document security. In this section, we will survey the correlation technique of controlling users access information based on data processing and identity verification, and then briefly state the advantages and challenges of these technologies.

Due to the explosive growth of data, most users store data in the cloud or public organizations. For this application environment, Shen et al. [16] proposed a remote data integrity audit scheme based on identity encryption, so that documents stored in the cloud can be shared and used by others when sensitive information is hidden, while remote data integrity audits can be effectively performed. In data processing, security and confidentiality are important for database administrators and users. Segundo et al. [17] proposed a hybrid blockchain model, solving the problem that the secu-

rity model suffered computer attacks because of security management vulnerabilities.

Personally identifiable information (PII) is served as a gateway to personal and organizational information when performing identity verification. Rima et al. [18] evaluated different *PII* options for user authentication in blockchain-based solutions, helped users make informed decisions based on the identity ecosystem model, and encouraged providers to introduce better choices to users. Wang et al. [19] designed a practical authentication scheme for mobile devices, which solved the problem of user authentication in mobile networks. An identity-based cryptosystem means that the public key could be obtained from the user identifier. Lin et al. [20] constructed a linear homomorphic signature scheme based on newID; they used a random oracle model to perform linear calculations on authenticated data, and effectively prevented the forgery of messages and IDs by avoiding the use of public-key certificates.

The above current technologies can prevent user identity from being maliciously tampered and ensure data security, but could not guarantee the authenticity and objectivity of data or achieve the purpose of traceability of historical data. From the perspective of data management, blockchain could be regarded as a tamper-resistant ledger maintained by many untrusted nodes in a distributed environment [21]. Its characteristics generally include decentralization, trustlessness, publicity and transparency, tamper-resistant, and anonymity [22].

The blockchain realizes the credible sharing of data among the parties that distrust each other, and the smart contract running on the blockchain realizes the credible execution of business logic [23]. In 1994, Szabo N first proposed the concept of smart contracts [24]. The essence of the smart contract is a modular, reusable, and automatically executed script code that runs on the network, allowing developers to personalize development based on specific protocols, businesses, or logic. And the purpose of a smart contract is to provide a safer method and reduce transaction costs [25,26]. The blockchain provides a good platform for the application of smart contracts because of its decentralization and distributed features. However, due to limited capabilities, insensitive to information changes of smart contracts, there are many difficulties when it is used as a security mechanism in a security scenario. Fan et al. [27] classified smart contracts according to the operating environment, discussed the application scope of their operating platforms, and analyzed the current situation and challenges of smart contracts in terms of security, scalability, and maintainability in-depth. To promote multi-user collaboration and trace document modification records in a trusted and secure way, Nizamuddin et al. [28] took advantage of PFS to store documents on the decentralized system. At present, the application scenarios of smart contracts are not enough, more in-depth discussion and analysis are needed for the challenges faced by the smart contract, as well as its security [29,30], scalability [31], and maintainability.

The existing studies seldom mentioned document encryption, authority authentication, and historical records trace, which makes it impossible to effectively guarantee the real-time sharing of documents and the authenticity and objectivity of data. In response to the above problems, we proposed a trusted verification scheme for office documents based on blockchain. The scheme discusses the authenticity and objectivity of data from the perspective of document encryption and authority authentication. We use blockchain to store users and documents information, formulate the conditions for triggering the smart contract, and design reasonable document uploading and monitoring process. Finally, the scheme can monitor the state change of office documents in real time, and achieve the purpose of document encryption and authority authentication.

Overall model

Related concepts

Blockchain is a new application pattern of computer technologies, as a distributed shared ledger, it has the features of decentralization, tamper-resistant, maintain collectively, transparency, traceability of throughout, and data. It uses the core technologies, including distributed storage, peer-to-peer transferring, consensus mechanism, and encryption algorithm, to provide participants with a credible, reliable, and transparent logic framework. The block-chain structure diagram is shown in Fig. 1.

Blockchain—The blockchain is a sequence of blocks, which holds a complete list of transaction records. The block, composed of header and body, is used to store transaction summaries, and it is also a structural unit of data storage

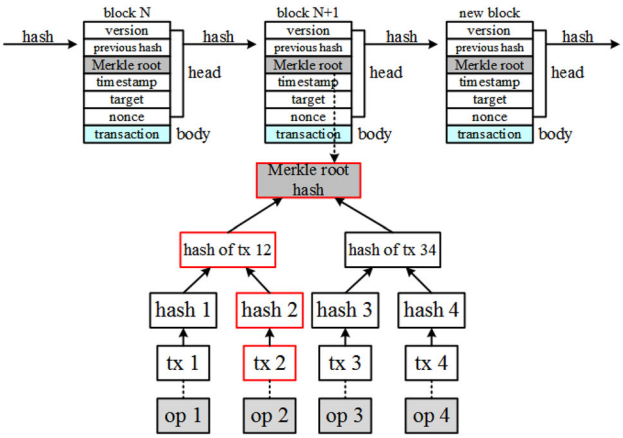


Fig. 1 Detailed blockchain structure

in the blockchain. The header includes the version number, hash of the previous block, root, and timestamp. The body is composed of transactions and counters; it is mainly used to store transaction summaries and verify transactions. The chain structure is formed by connecting each block through hash pointers in the order of generation time. The timestamp is used as proof of existence and notarized; it will prove that unauthorized modification could be detected under this condition as long as the document exists.

Smart Contract—The smart contract, or chaincode, is a piece of code written on the blockchain, which is deployed in the blockchain to describe related business logic. The deployed smart contract is unmodifiable in the blockchain, and its execution is completely determined by the code and not interfered with by human factors. It will periodically check whether the related events and trigger conditions exist; the events that meet the conditions will be pushed to the verification queue. The verification nodes on the blockchain will first perform a signature on the event to ensure its validity, we can successfully execute the smart contract and notify users after the nodes reaching consensus. Generally speaking, participants use smart contracts to specify their rights and obligations, conditions, and results that trigger the contract. Once the smart contract runs in the blockchain, objective and accurate results can be obtained.

The blockchain guarantees the authenticity of the authentication process, and the smart contract ensures that the certified contract is executed objectively and compulsively without any interference from external factors. Authority authentication and document encryption are realized through blockchain and customizing smart contracts, and the authenticity and objectivity of data are effectively ensured.

Model introduction

In this manuscript, we propose the TVS for office documents based on the features of blockchain, such as security, credibility, immutability, and traceability of network behavior. The scheme monitors the state changes of office documents by setting the trigger conditions of smart contracts, and it uses blockchain to store users and documents information in real time.

The overall model of the TVS for office documents is shown in Fig. 2; it includes the following modules:

- **Node network**—Establish a complete blockchain system node network topology.
- **Information initialization**—Provide a method to create and record users' access information.
- **Information query**—Provide methods for querying users and documents information and verifying correctness.

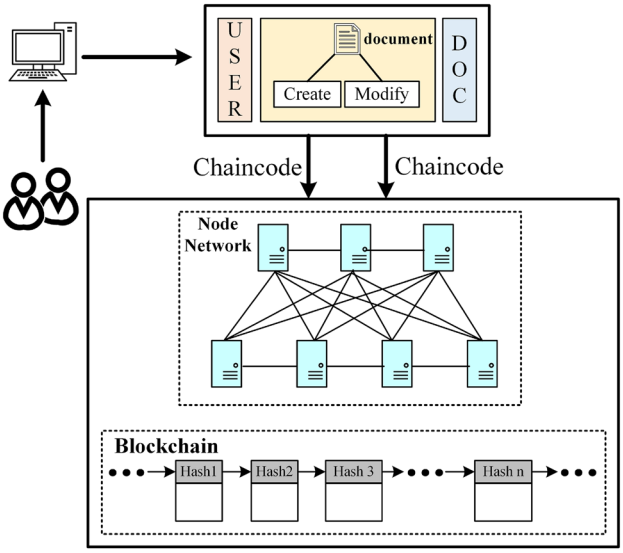


Fig. 2 The overall model of the TVS for office documents based on blockchain

- **Information modification**—Provide a method for modifying the document contents of certain users.

Function modules

To precisely describe the technical solution proposed in the manuscript, the main function modules will be described in detail below.

Node network

The node network module nodes users, digitizes users, and documents information and divides users into groups by category. Each user acts as a peer in the network; they supervise and cooperate to establish a complete blockchain system node network topology. The system includes multiple nodes; each node stores complete data and establishes a communication connection between each other. These nodes include the endorser, the orderer, the committer, and the certificate. The endorser and the committer are peers in the decentralized blockchain network. The responsibility of the endorser is to verify the transaction plan, simulate execution, and endorse. The orderer sorts the transactions sent by each node, determines the transaction sequence and reaches a consensus, sends the result back, and saves it in the ledger. The committer checks the legitimacy of transactions, and updates and maintains the data and ledger status of the blockchain. The certificate is responsible for providing system members with identity based on digital certificates, and realizing the authority control of the blockchain system on the basis of clear membership.

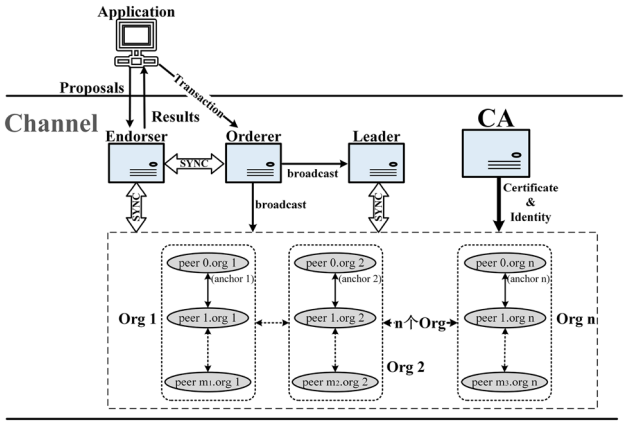


Fig. 3 Detailed topology of the node network module

Algorithm 1 Build node network

Procedure: SETUP($channel, orderer, peers, chaincode$)
1: $cryptogen$ generate;
2: $configtxgen$ generate;
3: $CHANNEL\_NAME \leftarrow mychannel$;
4: $path \leftarrow$ -p chaincode_example02/word/;
5: start networking: $docker - compose - cli.yaml$ up;
6: access CLI: $docker\ exec - it\ cli\ bash$;
7: $mycc \leftarrow mychannel$;
8: $peer0.org1$ create $mychannel$;
9: $peer0.org1$ join channel;
10: $peer0.org1$ update channel;
11: $peer0.org2$ repeat the procedure 9-10;
12: install chaincode on each peer;
13: instantiate the chaincode on $peer0.org1$.

The topology of the node network is shown in Fig. 3. In this module, different nodes belong to different organizations, forming a peer-to-peer decentralized network. Each organization has its clients, peers, and CA, it can create one or more different types of nodes as needed. The orderer is a component maintained by all nodes in the organization. In addition, we have designed a channel to provide privacy protection for members. The channel is an independent communication way between network members, and only the members belong to the channel could see the transactions sent in it. Each channel has its member organization, the anchor, and the orderer, and each organization has multiple nodes joining the same channel, and the peer0 is the anchor by default. The anchor conducts node interactions between organizations to discover all nodes in the channel. Besides, each channel elects or appoints a leader, which is responsible for receiving blocks sent from the orderer, and then forwarding them to other nodes in the organization. The leader ensures that the entire network could be maintained stability even the number of nodes is constantly changing.

In this manuscript, user nodes are added to the node network module by creating a channel. Multiple nodes in the channel jointly maintain a ledger, thus constructing a complete node network topology of the blockchain system. The procedure of building a node network is detailed in Algorithm 4.1.

Initialization

The initialization module provides method for creating and recording users and documents information. Users information refer to userID and individual user profiles. Documents information includes title and content. The above records form a blockchain transaction, which can generate a hash. And a new block could be generated using the block hash, block identifier, and timestamp. We define the initialization of information as an event; the smart contract will discover the event and automatically execute the storage request. And then, there will generate a new block added to the blockchain. The smart contract will periodically check whether the related events and trigger conditions exist. If reach the conditions, the event will be pushed to the queue to be verified. In the mean time, the verification node will first perform signature verification on the event to ensure its validity, the smart contract will be executed, and the user will be notified when most verification nodes reach a consensus. The basic verification and consensus process is shown in Fig. 4.

Through the above operations, each newly created record is hashed and stored in the blockchain. Meanwhile, the operations create a specific space for the hashed document and store it combined with the timestamp. This can protect not only document privacy and data security but also save data space and transaction overhead. Once the transaction

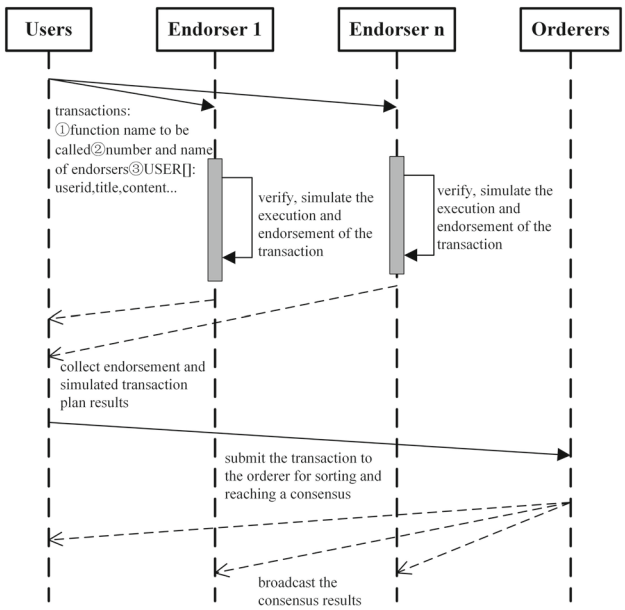


Fig. 4 Verification phase sequence diagram

is completed, there will return a transactionID, including the transaction hash, block number, and timestamp, which is used to verify the procedure of specific operation. Since the smaller hash is stored in the blockchain, it will take up little time during retrieval, so the verification could be completed in a few steps. In short, the initialization module can quickly realize storage and verification, and effectively guarantee the authenticity and objectivity of data. The algorithm used for initializing information is detailed in Algorithm 4.2.

Algorithm 2 Initialization

Input: Function name to be called
Input: Number and name of endorsers
Input: $USER[]$, the sets of users and documents information
function ININTLEGDER($args[]$)
  $i \leftarrow 0$;
  for $i <$ length of $args[]$ **do**
    $userAsBytes \leftarrow$ json.Marshal($user[i]$);
    PutState($strconv.Itoa(i)$, $userAsBytes$);
    Println ledger record;
    $i \leftarrow i + 1$;
  end for
end function

function CREATEUSANDDOC($args[]$)
  if length of $args[]$!= 5 **then**
    return $ERROR$;
  end if
  $startKey \leftarrow$ "$USER0$";
  $endKey \leftarrow$ "$USER999$";
  $resultsIterator \leftarrow$ GetStateByRange($startKey$, $endKey$);
  for the next set exist **do**
    $queryResponse \leftarrow$ resultsIterator.Next();
    if $arg[0] ==$ string($queryResponse.Key$) **then**
      return "The user is already exist!";
    end if
  end for
  $userAsBytes \leftarrow$ json.Marshal($user$);
  PutState($args[0]$, $userAsBytes$);
  return $SUCCESS$;
end function

Query

The query module provides method for querying and tracing historical records based on the traceability of blockchain. The information can be queried users information, document title, and content. The decentralized feature of the blockchain makes the records stored without a central node. We see all participating devices as nodes; each node is stored independently and has the same status. All nodes rely on a consensus mechanism to ensure the consistency of storage, while there is no single-node record or modifies data individually. These settings ensure data security and greatly avoid information loss caused by a single equipment failure.

When users create, modify, or query information on office equipment, the system can monitor the office document changes in real time and record each status completely. Meanwhile, the system links the generated blocks in chronological order based on the chain structure. In this way, the entire network data are backed up to provide a complete record of users and documents information. The query module can make users query documents at any time and realize the traceability and auditability of historical data. The algorithm used for querying is detailed in Algorithm 4.3.

Algorithm 3 Query

Input: Function name to be called
Input: The userid to be queried
function QUERYALLUSERS($args[]$)
  $startKey \leftarrow$ "$USER0$";
  $endKey \leftarrow$ "$USER999$";
  $resultsIterator \leftarrow$ GetStateByRange($startKey$, $endKey$);
  for the next set exist **do**
    $queryResponse \leftarrow$ resultsIterator.Next();
    if information exist in the ledger **then**
      buffer.WriteString($queryResponse.Key$);
      buffer.WriteString($queryResponse.Value$);
    end if
  end for
  Println all information of users and documents;
  return $SUCCESS$;
end function

function QUERYUSERANDDOC($args[]$)
  if length of $args[]$!= 1 **then**
    return $ERROR$;
  end if
  $startKey \leftarrow$ "$USER0$";
  $endKey \leftarrow$ "$USER999$";
  $resultsIterator \leftarrow$ GetStateByRange($startKey$, $endKey$);
  for the next set exist **do**
    $queryResponse \leftarrow$ resultsIterator.Next();
    if $arg[0] ==$ string($queryResponse.Key$) **then**
      $userAsBytes \leftarrow$ GetState($args[0]$);
      Println all information of users and documents;
      return $SUCCESS$;
    end if
  end for
  return "The userid not found!";
end function

Modification

The modification module provides method for modifying documents based on the tamper-resistant feature of the blockchain. The information can be modified includes the document name, title, and content of a certain user. Since each block is linked by a hash pointer, which is obtained by hashing the previous hash value and the block body, it will change once there is a data change. Therefore, the system realizes the modification by adding a block at the end of the

blockchain, thereby the historical data are still stored. We can view the historical records from logs. The above process ensures that documents could not be illegally tampered with based on the premise of traceable historical documents. The algorithm used for modifying is detailed in Algorithm 4.4.

Algorithm 4 Modification

Input: Function name to be called
Input: Number and name of endorsers
Input: $args[]$, the sets of users and documents information
1: **function** SETDOCUMENT($args[]$)
2: **if** length of $args[]$!= 5 **then**
3: **return** $ERROR$;
4: **end if**
5: $startKey \leftarrow$ "$USER0$";
6: $endKey \leftarrow$ "$USER999$";
7: $resultsIterator \leftarrow$ GetStateByRange($startKey, endKey$);
8: **if** err != nil **then**
9: **return** $ERROR$;
10: **end if**
11: **for** the next set exist **do**
12: $queryResponse \leftarrow$ resultsIterator.Next();
13: **if** $arg[0] ==$ string($queryResponse.Key$) **then**
14: **return** $ERROR$;
15: $userAsBytes \leftarrow$ json.Marshal($user$);
16: PutState($args[0], userAsBytes$);
17: **return** $SUCCESS$;
18: **end if**
19: **end for**
20: **return** "The userid not found!";
21: **end function**

In the above function modules, the blockchain is responsible for saving user personal information, document content, and other data. Every initialization, query, and modification operation of the user will be regarded as a transaction. After we digitize the operation, the blockchain system will save it to the root of the Merkle tree by calculating the hash value and adding it to the block header of the blockchain. The specific block content is shown in Fig. 5.

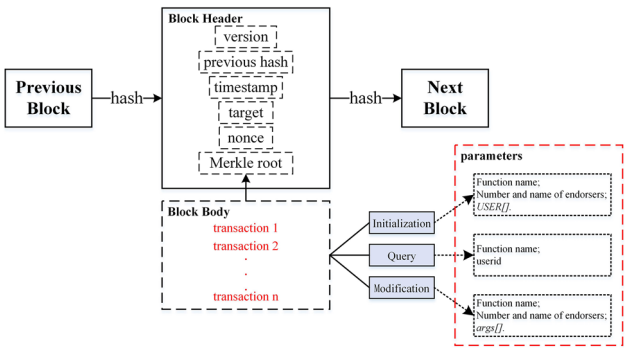


Fig. 5 The content of block

Experiment and evaluation

The *Hyperledger* project was initiated by the Linux Foundation in 2015; the institutions and companies involved include financial industry, manufacturing business, and logistics industry. The *Hyperledger Fabric* blockchain platform is a distributed ledger framework for enterprise solutions and applications [32,33]. *Fabric* is based on a modular design; it provides the architecture that in the order of executing first, sorting later, and verifying again. Nodes are divided into endorser, orderer, and committer, the transactions are executed on trusted endorsers and the results are propagated to all nodes to achieve a consistent state. The smart contract runs in parallel on different endorsers to improve the transaction throughput of the system [34,35]. In this manuscript, we use the *Hyperledger Fabric* as the experimental platform to conduct simulation experiments. During the experimental test, we set the consensus schema to Solo, which is a single-node communication mode of the orderer node, and it has only one orderer serving all nodes. Although the Solo schema has no high availability and scalability, and not suitable for the production scenarios, it is completely feasible to use in the development and test. The Gossip protocol has been used to achieve consensus among blocks in the Blockchain. At the same time, we use the Gossip protocol to implement algorithms that can finally make ledgers of different nodes in the organization consistent. Below will introduce the simulation experiments from four aspects, including build environment, system configuration, deploy chaincode, testing, and evaluation.

Build environment

To successfully build the *Hyperledger Fabric* platform, we need to prepare the runtime environment of the system and install the software for *Fabric* according to the fundamentality and dependency. The system uses the *Hyperledger Fabric v1.4.0*. We pre-install the environment dependencies according to the requirements, as shown in Fig. 6.

Due to the operating habits of *Ubuntu*, the installation of related software is performed at the command line; we rely on shell scripts during installation, call each other between multiple scripts, and interact with users as far as possible. After the dependencies are installed, we switch the operating directory to the *first-network* folder, and run the *byfn.sh* to start the fabric network. If the interface is shown as Fig. 7a, b, it indicates that the *Hyperledger Fabric* environment is set up and the fabric network is started successfully.

System configuration

Building environment is based on a series of configuration files, including the configuration file for generating cer-

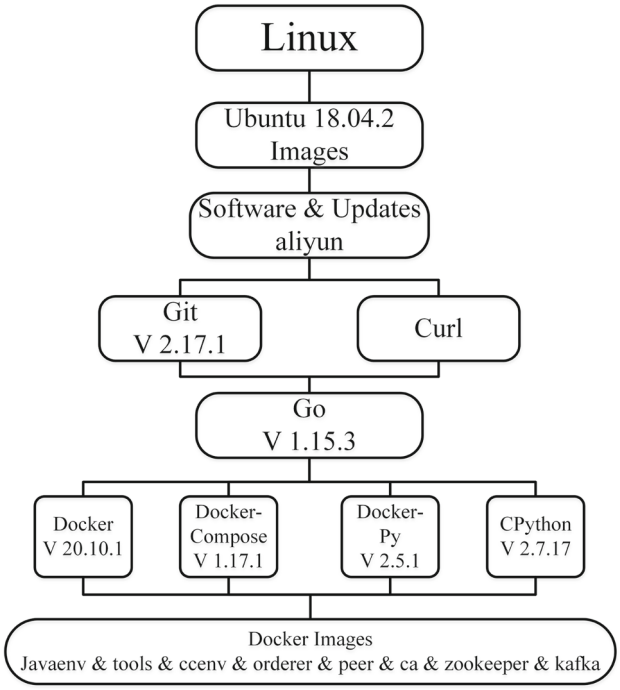


Fig. 6 The environment dependencies that need to be deployed to install *Hyperledger Fabric* in *Ubuntu*

tificates, the system genesis block, blockchain nodes, and databases.

Crypto-config.yaml

The cryptogen tool generates encrypted materials for network entities and uses certificates to represent various identities. The *crypto-config.yaml* is used in conjunction with the *configtx.yaml*. It is used to generate and verify certificates

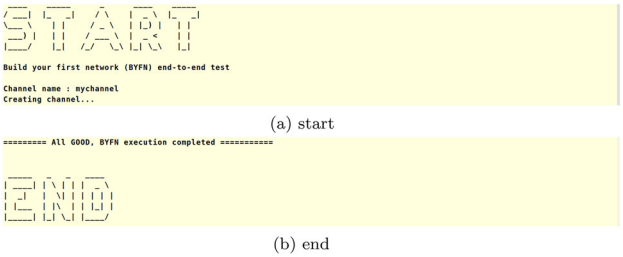


Fig. 7 The first-network start interface. **a** Interface that just started. **b** Interface that successfully started

and keys and further set user-defined organizations. The configurations include organization name, domain name, CA settings, and the number of nodes under the organization. The parameter settings of the *crypto-config.yaml* are shown in Table 1.

Configtx.yaml

The configtxgen tool is used to create four network artifacts, including *genesis.block*, *channel.tx*, *Org1MSPanchors.tx*, and *Org2MSPanchors.tx* [32]. The parameter settings of the *configtx.yaml* are shown in Table 2.

Orderer and peer

The orderer configuration file defines basic information, including the name, environment parameters, working directory, and ports of the orderer. The peer configuration file defines the general configuration of the endorser and parameter settings of the dockers. The parameter settings of the orderer and the peer configuration files are shown in Table 3.

Table 1 *Crypto-config.yaml* parameter configuration

Name	Main effect	Settings
OrdererOrgs	Define the management organizations of the orderer	Name:*Orderer* Domain:*example.com* Specs:*orderer*
PeerOrgs	Define the management organizations of peers	**Org 1:** Domain:*org*1.*example.com* EnableNodeOUs:*true* Template:2 Users:1 **Org 2:** Domain:*org*2.*example.com* EnableNodeOUs:*true* Template:2 Users:1

Table 2 *Configtx.yaml* parameter configuration

Name	Main effect	Settings
Profiles	Define the parms of the configtxgen, determine the genesis block and initial transaction info	$channelID$
		Organizations: $org1\&org2$
		Consensus: $solo$
Capabilties	Define the orderer and endorser capabilities range	Channel:$true$
		Orderer:$true$
		Application: $fabric\ v1.4-true$
Application	Define the genesis block read & write strategy	Reader:ANY
		Writer:ANY
		Admain: $MAJORITY$
Orderer	Define the parms of orderer written in the genesis block, read & write strategies	Domain: $example.com$: 7050
		BatchTimeout:$2S$
		AbsoluteMaxBytes:$99MB$
Organizations	Define orgsID referred in other parms	orgsID:$Org1MSP\&Org2MSP$
		AnchorPeers:$peer0$
		Policies:OR

Table 3 Orderer and peer parameter configuration

Name	Main effect	Settings
Orderer	Define the name, environment parms, working_dir and ports of orderer	**TLS:**
		PrivateKey:$tls/server.key$
		Certificate:$tls/server.crt$
		RootCAs:$-tls/ca.crt$
		Cluster:
		DialTimeout:$5s$
		RPCTimeout:$7s$
		ReplicationBufferSize:$20MB$
		ReplicationPullTimeout:$5s$
		ReplicationRetryTimeout:$5s$
Peer	Define working_dir, environment parms and dockers	**image:**
		hyperledger/fabric-peer: $\$IMAGE\_TAG$
		environment:$server.crt\&server.key\&ca.crt$
		working_dir:$/opt/gopath/src/github.com/hyperledger/fabric/peer$
		command: peer node start

Deploy chaincode

Firstly, we formulate a reasonable chaincode according to user requirements, including conditions for initializing users, and creating, saving, and accessing documents. Given initialization conditions, after the chaincode is uploaded, users' numbers and permissions are set by calling the initialization function. Given creation conditions, the system will automatically trigger the creation function. The contents to be stored are documents, users' access information, and block location. The storage request will be automatically executed after the chaincode detecting it; this operation will create a block and send the hash of records to the blockchain. Given query conditions, when the chaincode is formulated, the users obtained the documents will be set in advance by setting permissions of users. The system outputs data according to the constraint of the chaincode, transmit the data to the corresponding users, and further complete user access to documents. The procedure of chaincode deployment is shown in Fig. 8.

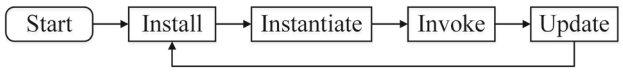


Fig. 8 Chaincode deployment procedure

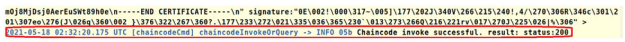


Fig. 9 The user and document are initialized successfully, and the success status is returned

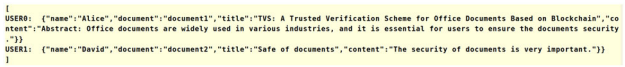


Fig. 10 When querying all users information, all records in the ledger are returned

After the chaincode is installed, instantiated, and invoked, it can realize functions such as initialization, query, creation, and modification. However, it should be noted that, when a new chaincode is uploaded, the latest code needs to be reinstalled, renamed, and instantiated on the nodes of the channel, and then realizes the function. In this way, the source chaincode is still stored in the blockchain even if it is modified.

Testing and evaluation

Next, we will test the system functions, including initialization of users and documents information query all information, and create and modify documents of a certain user.

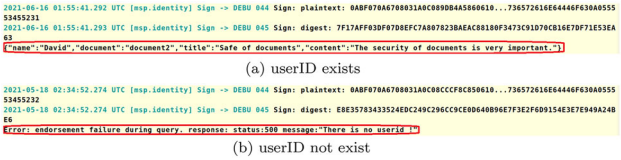
When the users and documents information is initialized, we need to call the initialization function, indicating the channel name, root certificate file, and the address of the peer that needs to be connected. The specific input as follows:

peer chaincode invoke -o orderer.example.com:7050 –tls –cafile $ORDERER_CA -C $CHANNEL_NAME -n mycc –peerAddresses peer0.org1.example.com:7051 –tlsRootCertFiles $PEER0_ORG1_CA –peerAddresses peer0.org2.example.com:7051 –tlsRootCertFiles $PEER0_ORG2_CA -c ’{“Args”:[“initLedger”]}’. The initialization result is shown in Fig. 9.

After initializing the users and documents information, the operation is recorded in the ledger, and we can output the information by calling the query function. We need to indicate the chaincode name, channel name, and the function name. The specific input as follows:

peer chaincode query -C $CHANNEL_NAME -n mycc -c ’{“Args”:[“queryAllUsers”]}’. The output result of the ledger is shown in Fig. 10.

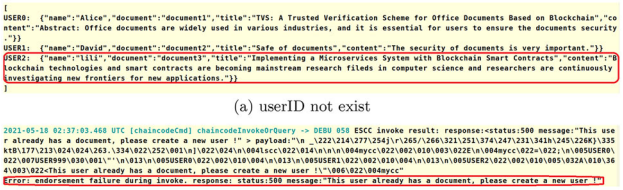
Similarly, when querying a certain user documents in the system ledger, it is necessary to indicate the channel name, the chaincode name, and the certain userID to call the query function. There will be two different situations, one is that the userID exists, all the documents information of the users



(a) userID exists

(b) userID not exist

Fig. 11 Query record for a certain user. **a** Query the document of USER1 and output certain record. **b** Queried userID not exist, prompting an error



(a) userID not exist

(b) userID exist

Fig. 12 Create new document record. **a** New document record could be displayed in the system ledger. **b** Newly created userID already exists, prompting an error

is printed, shown as Fig. 11a, the other is that the userID not exist, the error prompt is printed, shown as Fig. 11b. The specific input is as follows:

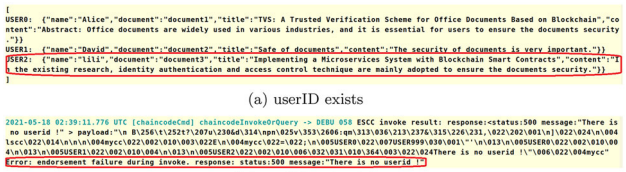
peer chaincode query -C $CHANNEL_NAME -n mycc -c ’{“Args”:[“queryUserandDoc”, “USER1”]}’.

Since the new users and documents information is written in the blockchain, in addition to the channel name, the chaincode name, and the address of the endorser, the root certificate file also should be indicated. There will be two different situations, one is that the userID not exist and the new document information is written into the ledger, shown as Fig. 12a; the other is that the userID already exists, and an error message is printed, shown as Fig. 12b. The specific input as follows:

peer chaincode invoke -o orderer.example.com:7050 –tls –cafile $ORDERER_CA -C $CHANNEL_NAME -n mycc –peerAddresses peer0.org1.example.com:7051 –tlsRootCertFiles $PEER0_ORG1_CA –peerAddresses peer0.org2.example.com:7051 –tlsRootCertFiles $PEER0_ORG2_CA -c ’{“Args”:[“createUsandDoc”, “USER2”, “lili”, “document3”, “Implementing Smart Contracts”, “Blo-ckchain technologies ****** for new applications.”]}’.

The principle of modifying is similar to the creation. First, we need to read the information from the ledger to find whether the userID exists. If exists, call the modification function to write new record into the ledger, shown as Fig. 13a. The old records are still stored in the ledger, and historical records can be inquired from system logs. If not exist, an error message will be printed, shown as Fig. 13b. The specific input is as follows:

peer chaincode invoke -o orderer.example.com:7050 –tls –cafile $ORDERER_CA -C $CHANNEL_NAME -n mycc –peerAddresses peer0.org1.example.com:7051 –tlsR-



(a) userID exists

(b) userID not exist

Fig. 13 Modify certain document record. **a** Modify the document record by certain userID, the new could be displayed in the system ledger. **b** The userID not exist, prompting an error

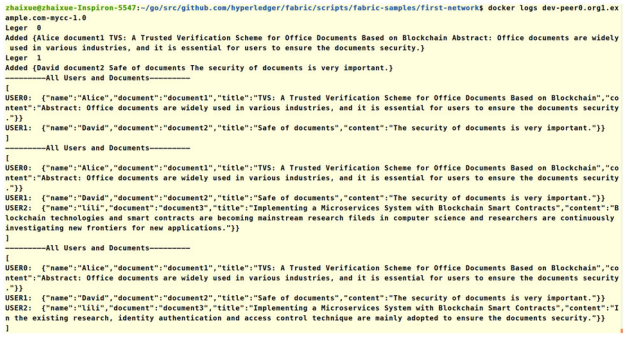


Fig. 14 System logs

ootCertFiles $PEER0_ORG1_CA –peerAddresses peer0.org2.example.com:7051 –tlsRootCertFiles $PEER0_ORG2_CA -c '{"Args":["setDocuments", "USER2", "lili", "document3", "Implementing ****** Smart Contracts", "In the existing ****** document security."]}'.

Based on the features of blockchain, a series of operations on users and documents information in the system can be queried. We can view the historical records from the ledger in system logs. Some logs are shown in Fig. 14.

In summary, all functions of the system could be successfully implemented. The evaluation is shown in Table 4.

After analyzing the experimental results, the proposed model has high security and can effectively prevent documents from being tampered. Specifically, when generating new blocks, as shown in Fig. 1, based on the generation principle of the Merkle tree and the irreversibility of hash operation, when the document information is modified, it is necessary to save the new information in the new block instead of directly modifying the original records, so as to ensure that all historical records are recorded and traceable. Accordingly, as shown in Figs. 12 and 13, when creating and modifying the document information, all endorsers in the user group are required to verify and simulate the execution and endorsement of the transaction, so as to enable the mutual cooperation and supervision of peer nodes in the blockchain network, and ensure the identity authentication and authority control. And as shown in Fig. 14, based on the characteristics of blockchain, decentralized, non-tampering, and traceable history records, a series of operations for document information in the system can be queried in the system log. It can effectively ensure the security of document information.

Additionally, in the complexity analysis of the proposed method, the algorithm time complexity of document information initialization, query, and modification is $O(n)$. During the simulation experiment, we mainly focus on the query and modification of document content in the blockchain, and realize the corresponding functions through the formulated smart contract. The completion time of initialization in Algorithm 4.2 is mainly related to the number of document information keywords set, and the completion time of information query and modification in Algorithms 4.3 and 4.4 is mainly related to the number of users need to be queried.

Conclusion and future work

In this manuscript, we propose a trusted verification scheme for office documents based on blockchain. The scheme sets the trigger conditions for smart contracts and uses the blockchain to store users and documents information. Due to the data are stored in multiple distributed nodes, the decentralized feature of the blockchain enables each node to have a complete record. Therefore, the scheme can greatly avoid the information loss caused by a single equipment failure, and it also provides complete records for users and documents

Table 4 Evaluation of testing

Functions	Description	Results		Pass
initLedger	Initialize user and doc info	Initialize userID, user name, doc name, title and content		Yes
createUsandDoc	Create user and doc info according to userID	UserID exists	ERROR	Yes
		UserID not exist	Successed create	
queryAllusers	Query all info	Output all info		Yes
queryUsandDoc	Query info according to userID	UserID exists	Output info	Yes
		UserID not exist	ERROR	
setDocuments	Query info according to userID	UserID exists	Output info	Yes
		UserID not exist	ERROR	

information to achieve the purpose of post-audit and traceability. Meanwhile, the execution conditions of the smart contract will be triggered once users operate on the document, so that the smart contract is automatically executed. Consequently, the scheme can monitor the state change of office documents in real time, and achieve the purpose of ensuring the authenticity and objectivity of data.

In future work, we will implement the proposed scheme in the application platform of office documents to ensure the historical records' tracing. Through the application in specific scenarios, we will continue to optimize the relevant smart contracts, and compare the proposed approach with other schemes in terms of block generation time vs. number of blocks, computational cost vs. number of blocks, and verification time vs. number of users, to further improve the security of the scheme and the query efficiency of document information.

Funding The study was funded by The Tai Shan Industry Leading Talent Project (Grant number tscy20180416).

Declarations

Conflict of interest There is no declaration of conflict of interest by the authors.

References

1. Li XS Research on Authenticated Encryption Technology for Important Information Document System. PhD thesis, Guizhou University
2. Chowdhury M, Colman A, Kabir MA, Han J, Sarda P (2018) Blockchain as a notarization service for data sharing with personal data store. pp 1330–1335
3. Chao Y, Xu M, Si X, Li B (2017) Blockchain with accountable cp-abe: How to effectively protect the electronic documents. In: IEEE international conference on parallel & distributed systems
4. Zhu XX, Fan T (2019) Research on application of blockchain and identity-based cryptography. In: 2018 4TH international conference on environmental science and material application
5. Wang T, Wen-Ping MA, Luo W (2019) Information sharing and secure multi-party computing model based on blockchain. Computer confernce
6. Tian H, Wang J, Chang CC, Quan H (2020) Public auditing of log integrity for shared cloud storage systems via blockchain. Wirel Netw 1–16
7. Datta S, Sinha D (2021) Besddffs: blockchain and edgedrone based secured data delivery for forest fire surveillance. Peer-to-Peer Netw Appl 12:1–30
8. Qiao SB, Pang SC, Luo G, Pan SL, Chen TT, Lv ZH (2021) Flds: An intelligent feature learning detection system for visualizing medical images supporting fetal four-chamber views. IEEE J Biomed Health Inf 1
9. Qiao SB, Pang SC, Zhai X, Wang M, Cheng XC (2020) Human body multiple parts parsing for person reidentification based on exception. Int J Comput Intell Syst 14(1)
10. Qiao SB, Pang SC, Luo G, Pan SL, Yu ZC, Chen TT, Lv ZH (2022) Rlds: an explainable residual learning diagnosis system for fetal congenital heart disease. Future Gener Comput Syst 128: 205–218
11. Qiao SB, Pang SC, Wang M, Zhai X, Dai F (2021) Online video popularity regression prediction model with multichannel dynamic scheduling based on user behavior. Chin J Electron 30(5): 876–884
12. Zhu XX, Wang D (2019) Application of blockchain in document certification, asset trading and payment reconciliation. In: 2018 international symposium on power electronics and control engineering (ISPECE 2018)
13. Sharma B, Sekharan CN, Zuo (F) (2019) Merkle-tree based approach for ensuring integrity of electronic medical records. In: 2018 9th IEEE annual ubiquitous computing, electronics & mobile communication conference (UEMCON)
14. Jia X, Hu N, Su S, Yin S, Zhang C (2020) Irba: an identity-based cross-domain authentication scheme for the internet of things. Electronics 9(4):634
15. Krishnan Prabhakar, Jain Kurunandan, Achuthan Krishnashree, Buyya Rajkumar (2021) Software-defined security-by-contract for blockchain-enabled mud-aware industrial iot edge networks. IEEE Trans Industr Inf PP(99):1
16. Shen W, Qin J, Yu J, Hao R, Hu J (2018) Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans Inf Forensics Secur 14(2):331–346
17. Toapanta M, Quimis Oae, Mafla E, Maciel R (2020) Analysis for the evaluation and security management of a database in a public organization to mitigate cyber attacks. IEEE Access 8:18
18. Rana R, Zaeem RN, Barber KS (2019) An assessment of blockchain identity solutions: minimizing risk and liability of authentication
19. Ding W, Cheng H, He D, Ping W (2016) On the challenges in designing identity-based privacy-preserving authentication schemes for mobile devices. IEEE Syst J PP(1):1–10
20. Lin Q, Yan H, Huang Z, Chen W, Shen J, Tang Y (2018) An id-based linearly homomorphic signature scheme and its application in blockchain. IEEE Access 20632–20640 10.1109/ACCESS.2018.2809426
21. Zhang ZW, Wang GR, Xu JL, Du XY (2020) Survey on data management in blockchain systems. J Softw 31(09):283–305
22. Liu YZ, Liu JW, Zhang ZY, Tong-Ge XU, Hui YU (2019) Overview on blockchain consensus mechanisms. J Cryptologic Res 6(4): 395–432
23. Shao QF, Zhang Z, Zhu YC, Zhou AY (2019) Survey of enterprise blockchains. J Softw 30(9): 2571–2592
24. Szabo N (1997) Formalizing and securing relationship on public networks. First Monday 2(9)
25. Fang Y, Cong LH, Yang ZB (2019) Study on digital smart contract based on blockchian. Comput Syst Appl 028(009):225–231
26. Harris CG (2019) The risks and challenges of implementing ethereum smart contracts. In: 2019 IEEE international conference on blockchain and cryptocurrency (ICBC)
27. Fan JL, Li XH, Nie TZ, Yu G (2019) Survey on smart contract based on blockchain system. Comput Sci 46(11):7–16

28. Nizamuddin N, Salah K, Ajmal Azad M, Arshad J, Rehman MH (2019) Decentralized document version control using ethereum blockchain and ipfs. Comput Electric Eng 76:183–197
29. Atzei N, Bartoletti M, Cimoli T (2017) A survey of attacks on ethereum smart contracts (sok). In: International conference on principles of security & trust
30. Marino B, Juels A (2016) Setting standards for altering and undoing smart contracts. In: Rule technologies: research, tools, and applications
31. Shao QF, Jin CQ, Zhang Z, Qian WN, Zhou AY (2018) Blockchain: architecture and research progress. Chin J Comput 041(005):969–988
32. Chen SB, Zheng SH, Tong YJ (2020) The core technologies of hyperledger fabric. Publishing House of Electronics Industry, Beijing
33. Aste T, Tasca P, Matteo TD (2017) Blockchain technologies: the foreseeable impact on society and industry. Computer 50(9):18–28
34. Androulaki E, Manevich Y, Muralidharan S, Murthy C, Laventman G (2018) Hyperledger fabric: a distributed operating system for permissioned blockchains. In: the thirteenth EuroSys conference
35. Kapritsos M, Yang W, Quema V, Clement A, Dahlin M (2012) All about eve: execute-verify replication for multi-core servers. In: Usenix conference on operating systems design & implementation