IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 August 2023, accepted 30 August 2023, date of publication 6 September 2023, date of current version 12 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3311712

RESEARCH ARTICLE

A Secure, Reliable and Low-Cost Distributed Storage Scheme Based on Blockchain and IPFS for Firefighting IoT Data

LINA LI[1], DEZHENG JIN[2], TINGTING ZHANG[2], AND NIANFENG LI[1]

[1]College of Computer Science and Technology, Changchun University, Changchun 130022, China
[2]College of Cyber Security, Changchun University, Changchun 130022, China

Corresponding author: Nianfeng Li (linf@ccu.edu.cn)

This work was supported in part by the Natural Science Foundation of Jilin Province, China, under Grant YDZJ202101ZYTS191; in part by the National Natural Science Foundation of China under Grant 61772228; in part by the Jilin Scientific and Technological Development Program under Grant 20210201083GX; in part by the Industry Research Innovation Fund of the University, China, under Grant 2020HYB03002; and in part by the Jilin Medical and Health Industry Development Special Project under Grant 20230401104YY.

ABSTRACT In the fire scene investigation, the firefighting Internet of Things (IoT) data is the key electronic evidence for event analysis and responsibility determination. However, the traditional centralized storage method leads to data easy to be tampered with and damaged. To solve these problems, this paper designs and implements a secure, reliable and low-cost distributed firefighting IoT data storage scheme based on the Fabric framework, combining blockchain technology, Interplanetary File System (IPFS) and Practical Byzantine Fault Tolerance (PBFT) consensus algorithm to provide a strong support for fire accident traceability. This scheme mainly includes the storage model, key algorithms and Fabric construction and improvement. IPFS stores the complete firefighting IoT data, as the off-chain storage system of the blockchain, and the blockchain only stores the storage address (IPFS hash) of data returned by IPFS, thus reducing the storage space overhead of the blockchain and ensuring data security. Further, we adopt the Fabric framework as the blockchain platform for firefighting IoT data, and embed the PBFT consensus algorithm into the framework to ensure the reliability of consensus nodes in Fabric, thus improving the availability of the blockchain. In addition, we use the AES and RSA algorithms to ensure the security of firefighting IoT data storage and transmission. Through system analysis and experimental testing, the proposed scheme meets the need for secure storage and traceability of firefighting IoT data. Compared with the storage scheme using only blockchain, the blockchain combined with IPFS technology has advantages in storage space occupation, significantly improved throughput, and lower latency overhead. Meanwhile, compared with the official Fabric, the improved Fabric supports Byzantine fault tolerance and has better security.

INDEX TERMS Firefighting IoT data, blockchain, IPFS, secure storage system, PBFT consensus algorithm, hyperledger fabric.

I. INTRODUCTION

Nowadays, fire is still the most common and dangerous hazard to public life and property. In order to reduce the occurrence of fire, many public and private places are equipped with appropriate fire safety equipment, such as fire extinguishers, smoke sensors and video monitors, to monitor the fire and provide electronic evidence during fire incident investigation [1]. In the traditional fire monitoring system, the firefighting IoT [2] data is stored on the central server. Although cloud computing [3] and other technologies are used to achieve collaborative storage across multiple regions [4], the centralized data storage in the region is vulnerable to malicious tampering and destruction [5], which may interfere with the determination of fire responsibility. Therefore, the secure and reliable storage of firefighting IoT data is crucial.

There are two main data storage solutions for the storage of firefighting IoT data: (1) Traditional solutions, which store the fire sensor information and monitoring video together

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

to the central server [1], [6], [7]; (2) Blockchain-based solutions. Only the fire sensor information is stored in the blockchain, and the fire monitoring video is stored on the central server [8], [9]. The above schemes solve the storage of firefighting IoT data, especially the second solution can effectively ensure the security of sensor data. However, both of them store the surveillance video that occupies a large amount of space in the central server, which is costly to store and cannot prevent tampering and malicious deletion. Meanwhile, since the blockchain network is likely to be attacked, there are still potential security risks in the data storage and transmission of the blockchain. In addition, there is no specific user-oriented available system implemented. Therefore, it is necessary and urgent to design and implement a low-cost, secure and reliable storage solution for the firefighting IoT data.

The application of blockchain technology to IoT data storage has become a hot research topic. For example, [10] presents a model which converts transaction records and sensor data occurring on IoT into blockchain structure and stores them in Raspberry Pi 3. Reference [11] proposes a data-sharing scheme based on blockchain and identity authentication. Reference [12] stores IoT information in blockchain and proposes a compression-aware data compression and reconstruction method to improve the efficiency of IoT information storage. Reference [13] proposes a blockchain-based secure transmission and storage scheme for IoT data, which improves the security of data transmission and storage. Although blockchain effectively prevents data from malicious tampering and deletion, each node in the blockchain holds a ledger, resulting in the occupation of a large amount of storage space [14]. To solve the problem of blockchain storage scaling, many solutions based on off-chain [15] storage have been proposed. For example, off-chain storage using Distributed Hash Table (DHT) [16], [17], [18], IPFS-based off-chain storage [19], [20], [21], [22], [23]. IPFS is gaining more and more attention as a blockchain off-chain storage solution that both extend the storage space of the blockchain and ensures that files are not tampered with once they are stored.

Currently, no work has been done to use blockchain and IPFS together for firefighting IoT data storage, nor has it been implemented in specific blockchain framework deployment. The firefighting IoT data storage scheme combining blockchain and IPFS needs to solve the following three challenges: (1) How to design and implement a storage framework combining blockchain and IPFS to ensure secure and low-cost fire data storage; (2) The distributed blockchain network has the Byzantine Generals Problem. How to ensure the reliability of the firefighting Consortium blockchain network; (3) How to ensure the security of data transmission in the process of submitting and downloading the firefighting IoT data.

In order to achieve the secure, reliable and low-cost storage scheme of firefighting IoT data, we refer to previous work and are inspired to design a storage framework based on blockchain and IPFS, including storage models and key algorithms, and use RSA and AES [24] to ensure the security of data transmission. Meanwhile, we take Fabric [25] as the blockchain framework and improve the reliability of blockchain by adding the PBFT consensus algorithm to solve Byzantine fault tolerance, and design and implement a blockchain browser [26]. The contributions of our work are as follows.

1) To the best of knowledge, we are the first to propose a secure and low-cost data storage solution for the firefighting IoT based on blockchain and IPFS. In this scheme, the firefighting IoT data is stored in IPFS in a distributed manner, and the blockchain only stores the IPFS hash value of the data, which not only prevents malicious tampering and deletion of the firefighting IoT data, but also saves the storage space of the blockchain, thus improving the operational efficiency of the blockchain system.

2) Further, based on Fabric, we implement the firefighting IoT data storage system. Specifically, we improve the Fabric framework by embedding the PBFT consensus algorithm to ensure the reliability of consensus nodes in the Fabric. Meanwhile, IPFS storage algorithms and Fabric smart contract algorithms are designed, and the AES and RSA algorithms are adopted to encrypt the firefighting IoT data, the IPFS hash value and the AES key, respectively, to support the secure storage and transmission of firefighting IoT data.

3) We evaluate our storage framework through system analysis and simulation experiments. The system analysis proves the security and reliability of the framework, and the simulation experiment shows the storage advantages of our framework by comparing with the two storage frameworks. In addition, we test the system throughput and latency to verify that our proposed framework performs better when applied to firefighting IoT data storage. In addition, we test the system throughput and latency to verify that our proposed framework has better performance when applied to firefighting IoT data storage.

The rest of this paper is organized as follows. Section II describes the related work, Section III describes the preliminaries of our work, Section IV discusses the proposed framework, Section V shows the implementation details, Section VI is the performance analysis of the blockchain system, and Section VII concludes the paper.

II. RELATED WORKS

In this section, we review some existing research works related to the work of this paper, including blockchain-based firefighting data storage, joint data storage of blockchain and IPFS, and performance improvements for Fabric, and discuss the shortcomings of these works.

Blockchain technology has been introduced in some works to solve the problem of secure storage of firefighting IoT data. Reference [8] stores fire equipment information in blockchain and uses Fabric to implement the traceability function of fire equipment information. Reference [9] uses blockchain to store and pass sensor data to specific units

to quickly predict the path of fire points and stop fires, but does not store video surveillance data. Both solutions store fixed information or real-time data via blockchain, which leads to a large storage space requirement. Many researchers have conducted research on expanding the storage space of blockchain. LightChain, proposed in [27], extends the blockchain using DHT, where each node no longer needs to keep the complete ledger, resulting in a 66-fold reduction in storage space for nodes. A blockchain storage expansion model based on the Chinese residual theorem is proposed in [28], which reduces the storage consumption of nodes by adopting different storage strategies for blocks with different security. In the Bitcoin network, blockchain expansion uses the Simple Payment Verification (SPV) protocol [29], which is a light node solution. The above solutions mainly improve blockchain storage from the perspective of reducing duplicate storage of ledgers, which reduces the storage space to some extent but does not change the nature of blockchain storage data, and there is still a large storage requirement.

In recent years, more work has combined blockchain with IPFS, in which IPFS has expanded the storage space of blockchain. Reference [20] proposes an IPFS-based blockchain storage model that solves the problem of limiting nodes to join the network due to the gradual increase in the size of the Bitcoin ledger, and the security and synchronization speed of new nodes are also better. Reference [30] uses a ciphertext policy-based system to store encrypted electronic medical data in IPFS and utilizes blockchain technology to achieve secure storage and search of medical data in the storage platform. Reference [31] proposes a decentralized peer-to-peer image and video sharing platform based on IPFS and blockchain, uploads multimedia into IPFS and stores multimedia phash into the blockchain, and then uses phash technology to detect multimedia copyright infringement. Reference [32] proposes BlockMedCare, a secure healthcare system integrating IoT and blockchain, which uses IPFS as an off-chain database to store data, and then adopts smart contracts to control user access. Although the above work better solves the storage expansion problem of blockchain, there is no work on using IPFS as an off-chain storage expansion for firefighting IoT data.

Hyperledger Fabric is a mainstream blockchain platform, which has a large improvement in transaction processing speed compared to Bitcoin [29] or Ethereum [33] and provides enterprise-grade security. However, there are imperfections in Fabric functionality, and some work has optimized it. For example, [34] improves the transaction speed of Fabric to 20,000 transactions per second by optimizing the I/O, caching, parallelism, and data access parts of the Fabric transaction flow. Reference [35] proposes to apply BFT-SMART to the Fabric sorting process to support Byzantine fault tolerance, and optimizes the wide-area deployment to ensure that the blockchain system has fast transaction speed. Reference [36] uses gating and ring signature to hide the initiator in each consensus round, and divides the network into multiple minimum connection networks to reduce system overhead and improve scalability and security. Reference [37] proposes an efficient and low-cost dynamic Byzantine Fault Tolerance (BFT) algorithm. It handles member requests independently of view changes, and nodes can join and leave the system. Although the above work improves the performance and security of Fabric from different aspects, it is difficult to avoid affecting the stability of the system due to the large changes to Fabric.

III. PRELIMINARIES

We expound some background information in this section, including blockchain, IPFS and Fabric, which will be the basis of the proposed framework of firefighting IoT data storage system.

A. BLOCKCHAIN

Blockchain originates from the Bitcoin basic technology proposed by Nakamoto (pseudonym) in 2008 [29], which is a distributed ledger (database) maintained by multiple parties, and each party holds a complete ledger. Blockchain attacks need to primary at least half of the computing power of the entire network to achieve, thus ensuring the invariability of the ledger and the traceability of transaction records [38]. Blockchain consensus algorithm is one of the core technologies of blockchain, which solves the problem of consistency between nodes in distributed scenarios. Different consensus algorithms have different performance and security [39]. Consortium blockchain [40] is a blockchain composed of multiple trusted institutions, usually using the consensus algorithm of distributed systems. Smart contract [41] is a computer program first proposed by American computer scientist Nick Szabo in 1994. The contract terms in the blockchain are pre-defined and automatically executed by developers. Compared with traditional contracts, smart contracts have the advantages of trust removal, low cost, security and no third-party arbitration [42].

B. IPFS

IPFS [43] is a peer-to-peer distributed file system protocol that uses a new hypermedia transfer protocol based on content addressing, and its transfer mode is non-centralized peer-to-peer (P2P) mode, reducing the risk of attack on the central server. Meanwhile, when uploading files, IPFS calculates the file's hash value, which ensures that the file cannot be tampered with after uploading and is more suitable for storing important data. the core technology of IPFS mainly involves DHT [44], Merkle Directed Acyclic Graph (DAG), BitTorrent protocol [45], distributed version control Git [46] and Self-Certifying File System (SFS) [47]. Merkle DAG is a tree-like data structure used to ensure the integrity and verifiability of files stored in IPFS. In IPFS, a file is divided into several data blocks, and the hash value of each data block is stored on the leaf node of the Merkle DAG. The hash value of the parent node is calculated and generated by the hash value of its leaf node. Finally, the hash value

of the root node is the IPFS hash of the file. DHT is a distributed key-value storage system, which provides IPFS with powerful distributed storage and addressing capabilities and helps reduce the pressure of single point of failure on the central server.

C. FABRIC

Hyperledger Fabric [48] is an open-source, enterprise-oriented blockchain platform that not only has the decentralized, tamper-evident and traceable features of a blockchain, but also an enterprise-level access mechanism that blocks unauthorized access to the blockchain network and is suitable for storing sensitive data [49]. Fabric has a highly modular and configurable Fabric's smart contracts, called Chaincode, can execute relevant business code on the blockchain according to the logic pre-written by the developer [50], which currently supports Go language, Java language, and NodeJS language. We can also change the consensus algorithm and performance tuning of Fabric according to business requirements, which is more energy efficient compared to the Proof of Work (POW) algorithm used by Bitcoin [51], and more flexible in terms of block out time and block size. Due to performance issues, Hyperledger Fabric no longer supports PBFT algorithm after version 1.0. The existing consensus algorithms are Solo algorithm, Raft algorithm [52] and Kafka [53] algorithm.

IV. PROPOSED FRAMEWORK

A. THE SYSTEM MODEL

The blockchain and IPFS-based firefighting IoT data storage system model consists of four main parts: the firefighting IoT data acquisition module, the IPFS-based data storage module, the Fabric-based blockchain module and the browser-based firefighting IoT data query module, as shown in Figure 1. The data acquisition module is used to obtain video monitor data and various types of sensor data, such as smoke sensors, temperature sensors and water pressure sensors; the IPFS module stores firefighting IoT data; the blockchain module only stores the transmission records of firefighting IoT data; the query module supports users to query and track the transmission records of firefighting IoT data through the web page, and the browser data is stored in PostgreSQL. In this paper, we mainly focus on and describe the design and implementation of IPFS and blockchain modules.

In this model, the users of the system include departmental users and administrators, where the departmental users are responsible for collecting firefighting IoT data, referred to as users in the following, and the administrator's machine and the IPFS server each store a copy of the firefighting IoT data, the firefighting IoT data is collected by the hosts of each firefighting department, and the monitoring video and sensor data in JavaScript Object Notation (JSON) format are packaged into.zip format by using automatic scripts, and then sent to the web server through web pages. After receiving the data, the Web server uploads the data to the IPFS storage server, and calls the Fabric Software Development Kit (SDK) to upload the transmission record. Once the data is successfully uploaded, the system will return the success information to web pages, and users can view the transaction record in the blockchain browser to confirm whether the data is successfully uploaded. Then, the administrator can query the firefighting IoT data through the private key in the query interface, and fill in the encrypted IPFS address and his private key into the download interface to download the corresponding data. The process of storing and downloading firefighting IoT data is shown in Figure 2.

Algorithm 1 Users Upload Firefighting IoT Data to IPFS

Input: sender's private key, administrator's public key, firefighting IoT data

Output: IPFS Hash of encrypted firefighting IoT data, encrypted AES key

```
aes_key ← generate_random_aes_key()
enc_data ← aes_encript(data, aes_key)
enc_aes_key ← encript_with_rsa(aes_key, Admin_pk)
data.ipfs_hash ← ipfs.add(enc_data)
data.encrypted_ipfs_hash ← ursa.encrypt(data.ipfs_hash, Admin_pk)
if data.encrypted_ipfs_hash ≠ nil then
    return data.encrypted_ipfs_hash, enc_aes_key
return Error("upload err.")
```

B. KEY ALGORITHMS

1) IPFS STORAGE ALGORITHMS

In the framework, IPFS is used on the storage server to store the complete firefighting IoT data, thus expanding the storage space of the blockchain, and providing the blockchain with the address of firefighting IoT data in IPFS, namely IPFS hash. There are two IPFS offline storage algorithms: the firefighting IoT data upload algorithm and the firefighting Internet of Things data download algorithm, which ensure the storage and availability of firefighting IoT data.

Algorithm 1 (upload function) is used for users to upload firefighting IoT data to IPFS. First, the user generates a random AES key and encrypts the firefighting IoT data with it. Then the user uses the administrator's RSA public key to encrypt the AES key and destroys it after sending the encrypted key to the administrator via HyperText Transfer Protocol Secure (HTTPS). Next, the user fills in the private key, the administrator's public key and the encrypted firefighting IoT data on the web page, and submits the upload request to the web server. After receiving the user's request, the web server calls the ipfs-http-client library to store the encrypted firefighting IoT data to the IPFS server. Then, the IPFS server generates an IPFS hash and encrypts it with the administrator's public key. Finally, the encrypted IPFS hash and AES key are returned to the web server.

Algorithm 2 (download function) is the algorithm for the administrator to download the firefighting IoT data from

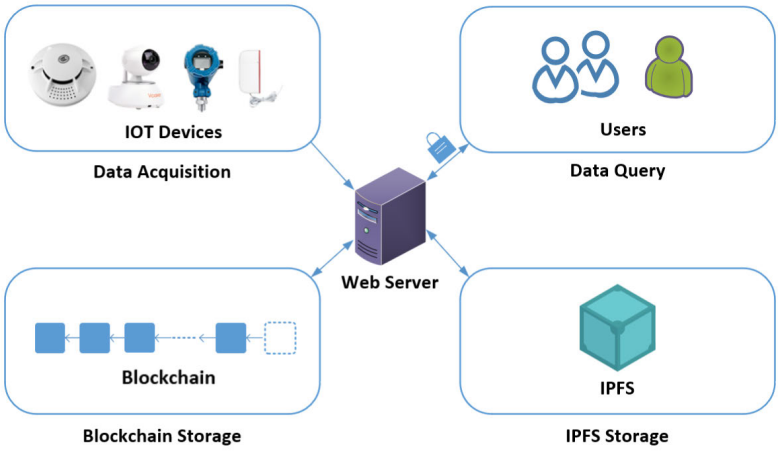

FIGURE 1. The model of firefighting IoT data storage system.

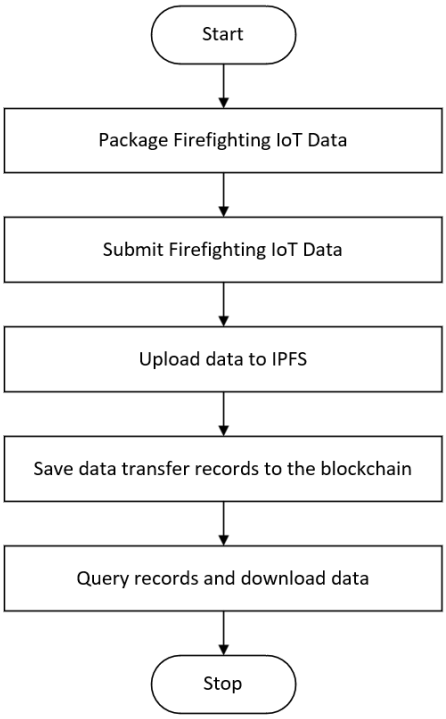

FIGURE 2. The storage and download process of firefighting IoT data.

IPFS. After querying the firefighting IoT data, the administrator fills in the private key and the encrypted IPFS hash of the firefighting IoT data in the web page, and submits the download request to the web server. Once the web server receives the download request, it uses the administrator's private key to decrypt the encrypted IPFS hash value of the encrypted firefighting IoT data, and calls the ipfs-http-client library to download the encrypted firefighting IoT data from the IPFS server and returns it to the administrator. The administrator uses the private key to decrypt the encrypted AES key, and then uses the AES key to decrypt the encrypted firefighting IoT data.

Algorithm 2 Administrators Download Firefighting IoT Data

Input: administrator's private key, encrypted IPFS Hash of firefighting IoT data, encrypted AES key

Output: firefighting IoT data

1: $data.ipfs\_hash \leftarrow$
$ursa.decrypt(data.encrypted\_ipfs\_hash, Admin\_sk)$
2: $enc\_data \leftarrow ipfs.download(data.ipfs\_hash)$
3: $aes\_key \leftarrow decrypt\_with\_rsa(enc\_aes\_key, Admin\_sk)$
4: $data \leftarrow aes\_decrypt(enc\_data, aes\_key)$
5: **return** *data*

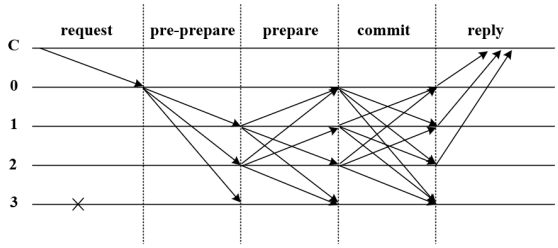

FIGURE 3. The PBFT consensus process.

2) PBFT CONSENSUS ALGORITHM

In our framework, due to hardware errors, network congestion or interruption and malicious attacks, the nodes in the system may become Byzantine nodes, thus affecting the safe operation of the blockchain network. In order to improve the security of the blockchain network, this paper embeds the PBFT consensus algorithm into Fabric. PBFT is a widely used distributed consensus algorithm proposed by Castro et al. [54] in 1999, which mainly solves the Byzantine general problem. The fault-tolerant number of the algorithm is $3f + 1 \leq n$, where f is the number of Byzantine nodes and n is the total number of nodes. The complexity of the PBFT algorithm is polynomial, which makes it feasible in practical applications.

Algorithm 3 Users Upload the Transmission Record

Input: sender's private key, administrator's public key, timestamp, encrypted IPFS Hash of firefighting IoT data
Output: uploading success or failure

1: $Sender\_pk \leftarrow ursa.getPublicKey(sender\_sk)$
2: $record \leftarrow$ $Sender\_pk, Admin\_pk, Ts, data.encrypted\_ipfs\_hash$
3: $flag \leftarrow APIstub.PutState(record)$
4: **if** $flag \neq nil$ **then**
5: **return** $Error(\text{``}stub.PutState\ err.\text{''})$
6: **return** $Success(\text{``}Upload\ success!\text{''})$

The PBFT algorithm needs five phases to reach a consensus, as shown in Figure 3, where "C" is the client, "0", "1", "2" and "3" represent four nodes, and the node with the symbol "x" is the fault node (Byzantine node). The details of the consensus are as follows. (1) request phase. The firefighting department user sends a request to the primary node of PBFT in the Fabric network through the web page; (2) pre-prepare phase. The orderer primary node in Fabric verifies the request message and broadcasts it to other nodes if it passes the verification; (3) prepare phase. PBFT node verifies whether the pre-prepare message is valid, and if it passes, it broadcasts the pre-prepare message to other nodes. After receiving $2f + 1$ prepare messages, the node broadcasts the commit message; (4) commit phase. After receiving $2f + 1$ valid commit confirmation messages, the node sends a reply message to the web server; (5) reply phase. When the web server receives $f + 1$ reply messages, the PBFT consensus node is considered to have completed consensus on the message.

3) SMART CONTRACT ALGORITHMS

In this framework, we design two smart contract algorithms: the uplink algorithm and the query algorithm, which store the transmission records of firefighting IoT data on the blockchain to ensure its tamper-proof and traceability, and can be queried.

Algorithm 3 (sendhash function) is the uplink algorithm of the transmission record of the firefighting IoT data. First, the user obtains the sender's public key from the sender's private key by using the ursa library. Then the user generates a transfer record of the firefighting IoT data into IPFS, which contains the sender's public key, the administrator's public key, the timestamp and the encrypted IPFS hash. Finally, the user calls the APIstub.PutState() function to upload the record to the blockchain.

Algorithm 4 (query function) is the algorithm for the administrator to query the firefighting IoT data. After receiving the firefighting IoT data, the administrator converts the private key into the public key through the ursa library on the web server, and then calls the query function in the smart contract with the public key to view the transmission records of all firefighting IoT data.

V. IMPLEMENTATION OF SYSTEMS

A. IPFS-BASED STORAGE

The firefighting IoT data is mainly divided into fire equipment sensor data and video monitoring data. The sensor data mainly contains water pressure, temperature and vibration of the pump, water pressure of the firefighting network, smoke sensor of the smoke alarm, temperature of the temperature sensor, fire door opening and closing situation and equipment location information, etc. In this system, sensor data is stored using JSON format, occupying a space of generally 1KB; video monitoring data is the 24-hour monitoring video of multiple cameras, taking the monitoring video of 1080P surveillance cameras as an example, the 24-hour monitoring video occupies a space of 50 GB or so.

In the firefighting IoT data storage system, the IPFS server stores files in chunks and each file is organized in the Merkle DAG format, using the root hash of the Merkle DAG (IPFS hash) to represent the file. The version of IPFS we use is go-ipfs_v0.4.19_linux-amd64, which is installed directly on the web server. When the IPFS server is started, the web server implements the function of uploading and downloading data by using the ipfs-http-client module, and the process is as follows.

(1) The web server initiates a request to upload firefighting IoT data using the ipfs.add function in the ipfs-http-client module, and the function parameter is the path to upload the firefighting IoT data.

(2) The IPFS server, after receiving the request about uploading data, splits the data into 256KB chunks (IPFS objects), then assigns a unique hash value (Content ID) to each object and organizes the objects using Merkle DAG format, and the root hash is the IPFS hash of the firefighting IoT data.

(3) The web server initiates a request to download the firefighting IoT data using the ipfs.get function in the ipfs-http-client module, with the function parameters being the IPFS hash of the data and the path to store it.

(4) After receiving the IPFS hash, the IPFS server finds the location of the firefighting IoT data through DHT and then downloads the firefighting IoT data from the node holding the data using the bitswap protocol.

(5) The data is returned to the user through the web server.

B. FABRIC-BASED BLOCKCHAIN

1) ADDING PBFT TO FABRIC FOR BLOCKCHAIN RELIABILITY

In this section, we describe the process of adding the PBFT consensus algorithm to Fabric 1.4.4. First, we need to make configtxgen recognize the PBFT consensus algorithm, then the orderer node runs the PBFT algorithm, and finally, just start the corresponding orderer container [55] when starting the blockchain network. The steps to add PBFT to the first network in Fabric 1.4.4 are as follows.

(1) Configure the basic environment, including installing gcc, make and g++ packages, the go language environment, and configuring GOPATH and GOROOT environment variables.

Algorithm 4 Administrators Query the Transmission Records

Input: administrator's private key
Output: all transmission records

```
Admin_pk ← ursa.getPublicKey(Admin_sk)
records, flag ← APIstub.GetState(Admin_pk)
if flag ≠ nil then
    return Error("stub.GetState err. ")
return records
```

(2) Copy the source code of PBFT consensus algorithm to Fabric source code orderer/consensus/ directory.

(3) Modify the Fabric source code, including config.go, encoder.go, and main.go files, to make it support PBFT, as shown in Table 1.

(4) Get the tools related to compiling the source code by cloning the github.com/golang/tools project to the $GOPATH/src/golang.org/x directory.

(5) Set GO111MODULE to off, and use the command "make" to compile configtxgen and orderer-docker to support the modification of PBFT consensus algorithm.

(6) Start first-network related files, mainly including: crypto-config.yaml, configtx.yaml, docker-compose-cli.yaml, byfn.sh, scripts/script.sh, scripts/utils.sh, docker-compose-base.yaml, peer-base.yaml. Then, modify the above files except for the crypto-config.yaml file. The profile is written in configtx.yaml to start PBFT, and refers to SampleMultiNodeEtcdRaft for details.

(7) Modify three scripts in sequence: byfn.sh, scripts/script.sh, and scripts/utils.sh, as shown in Table 2.

(8) Create the docker-compose-pbft.yaml file, add three orderer configuration files to the file, and keep the other contents the same as docker-compose-cli.yaml.

(9) Use the command "./byfn.sh up -o pbft" to start the first-network using the PBFT consensus algorithm.

2) BUILDING A BLOCKCHAIN NETWORK FOR FIREFIGHTING IoT

Fabric mainly consists of system module and tool module. The system module includes orderer and peer, in which the orderer node sequences and verifies transactions according to the consensus algorithm, while the peer node maintains the ledger and smart contracts (chaincode), handles SDK requests, and maintains the consistency of the ledger across multiple nodes. The tool modules include cryptogen, configtxgen and configtxlator, where cryptogen is used to generate cryptographic certificates and keys, and to generate certificates and keys for the organization according to the user's configuration requirements; configtxgen is used to generate the configuration files for the channel's creation blocks and channel transactions, which can be used to blockchain network for performance tuning, such as setting block size, generating block time, modifying consensus algorithm, and realizing tuning of upper layer applications of blockchain network; configtxlator mainly serves to convert some binary files that cannot be edited by users, such as.tx founding block files and.block files, into user-readable JSON files.

We use the above modules to build the blockchain network, which is mainly composed of Fabric Certificate Authority (CA), peer node and orderer node. Specifically, we first add the same channel to each peer node, and then install and instantiate the chaincode to implement the uplink of firefighting IoT data. In the firefighting IoT data blockchain, peer nodes can be enterprises, regulatory departments and other institutions. This paper aims to solve the problem of firefighting IoT data blockchain storage expansion. Therefore, the peer node is simplified into four departments of an enterprise. Each department has a server for saving the firefighting IoT data, which not only realizes the transparent sharing of information between departments, but also ensures the security of data. The peer node settings in the firefighting Fabric instance are shown in Table 3

In the firefighting blockchain network, the transaction consensus is used to ensure that the firefighting IoT data is not tampered with or deleted, and supports the query operation on the blockchain through the blockchain browser to confirm the transmission process of the firefighting IoT data on the blockchain. The basic steps to reach a consensus on the data transmission transaction of the firefighting IoT are shown in Figure 4.

(1) The web server invokes the certificate service through the SDK to communicate with the designated Fabric-CA server for registration and enrollment, and obtains the Identity Document (ID) certificate.

(2) The SDK of the web server sends a transaction proposal to the Endorser node of Fabric according to the API provided by the blockchain network, which contains the channel information, the ID of the invoked smart contract, timestamp, client signature, transaction content (chaincode functions and parameters).

(3) The Endorser node validates and simulates the execution (endorsement) of the received transaction proposal, and the contents of the validation are: whether the format of the proposal is correct, whether the transaction has been submitted (to prevent double flower attack), the client signature of the proposal, and the client authority. When the verification of the transaction proposal is passed, the simulation execution is carried out and the corresponding values generated after the execution, the results of the read-write set are endorsed and returned to the web server.

(4) After the web server collects enough information returned by the Endorser node, if it is querying the ledger information, the information will be returned directly; if the transaction modifies the ledger, a transaction request signed by the transaction proposal and the Endorser node will be sent to the Orderer node;

(5) Once receiving the transaction request, the Orderer node sorts the transaction request in chronological order, creates a transaction block, and then broadcasts it to the Leader nodes of all organizations in the same channel.

TABLE 1. Fabric source code modification details.

filename	line number	modification
common/tools/configtxgen/localconfig/config.go	388	Add “case "pbft":”
common/tools/configtxgen/encoder/encoder.go	39	Add “ConsensusTypePbft = "pbft"”
common/tools/configtxgen/encoder/encoder.go	216	Add “case ConsensusTypePbft:”
orderer/common/server/main.go	50	Add “"github.com/hyperledger/fabric/orderer/consensus/pbft"”
orderer/common/server/main.go	650	Add “consenters = pbft.New()”

TABLE 2. Fabric first-network scripts modification details.

filename	line number	modification
byfn.sh	171	Add “elif ; then COMPOSE_FILES="-f $COMPOSE_FILE_PBFT"”
byfn.sh	196	Add “if ; then sleep 1 echo "Sleeping 15s to allow $CONSENSUS_TYPE cluster to complete booting" sleep 14 fi”
byfn.sh	237	Add “elif ; then COMPOSE_FILES="-f $COMPOSE_FILE_PBFT"”
byfn.sh	442	Add “elif ; then configtxgen -profile SamplePbft -channelID $SYS_CHANNEL -outputBlock ./channel-artifacts/genesis.block”
byfn.sh	519	Add “COMPOSE_FILE_PBFT=docker-compose-pbft.yaml"
scripts/script.sh	/	Replace all "7050" with "6050"
script/utils.sh	/	Replace all "7050" with "6050"

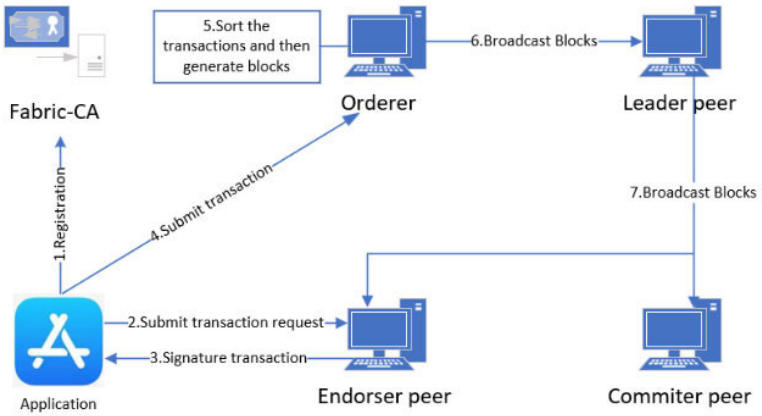

FIGURE 4. The data transmission transaction process of the firefighting IoT.

TABLE 3. Fabric peer node settings.

Participant	Organization	Node
Department 1	Org1	peer0.org1. example.com
Department 2	Org1	peer1.org1. example.com
Department 1	Org2	peer0.org2. example.com
Department 2	Org2	peer1.org2. example.com

(6) The Leader node verifies the block sent by the Orderer node after receiving it: read/write set version, transaction format, whether it is duplicated, whether it has sufficient endorsement, and if the block passes the verification, it writes the block to the local ledger.

(7) The Leader node broadcasts the validated block within the organization, and the anchor node is responsible for broadcasting the block across the organization. The peer node writes the received block to the ledger and notifies the web server that the transaction proposal has been written to the blockchain.

VI. EXPERIMENTAL EVALUATION

A. EXPERIMENTAL SETUP AND DEPLOYMENT

The firefighting IoT data storage system is built in the virtual machine Vmware workstation 16.0, and the main hardware and software information for the experimental simulation test is shown in Table 4. The system environment and deployment are as follows.

TABLE 4. Hardware and software specifications.

Name	Specification/Version
CPU	AMD R5 5600X 4 core
Disk	100G SSD
Memory	8GB
Operating System	Ubuntu 20.04
Docker	20.10.13
Docker-compose	2.2.2
Hyperledger Fabric	1.4.4
NodeJS	8.15
Sails.js	1.5.2
IPFS	0.4.19
Python	2.7

(1) The blockchain network of this system is the first-network in fabric-samples (version 1.4.4), which has a built-in byfn.sh script that can be started easily with the start command “byfn.sh up -n -o pbft”. In addition, the “docker ps” command allows us to check whether the network is running properly.

(2) Install the chaincode on the Endorser node. In this system, peer0 of organization 1 and peer0 of organization 2 are

FIGURE 5. An example of generating a key pair.

the Endorser nodes of Fabric, and the endorsement policy is ''OR''. Therefore, the chaincode needs to be installed and instantiated on these two nodes to support the application to initiate transaction requests.

(3) Start the IPFS server. The ''IPFS daemon &'' command is used to start the IPFS server to support the web server to send requests to the IPFS server.

(4) Start PostgreSQL. The ''docker-compose'' command is adopted to start PostgreSQL, the database container of the blockchain browser.

(5) Launch the Sails.js framework through the command ''sudo $HOME/.nvm/versions/node/v8.15.0/bin/node./node_modules/sails/bin/sails.js'' to support users to access the blockchain storage system through the browser.

B. ANALYSIS OF SECURITY AND RELIABILITY OF FIREFIGHTING IoT DATA

Firstly, in our scheme, the transmission records including IPFS hash of the firefighting IoT data are stored in the Fabric blockchain, ensuring that the firefighting IoT data can not be tampered with. On the one hand, Fabric has the general characteristics of blockchain, making it impossible to tamper with IPFS hashes stored in each block. Specifically, Fabric's ledger consists of two parts: world state and blockchain. The world state stores the current value of the Ledger status, which is expressed as a key value pair; The blockchain is a transaction log that records all changes in the current world state of construction. Once data is written to the blockchain, it cannot be tampered with. Specifically, in the structure of blockchain, blocks are composed of three parts, namely block headers, block data, and block metadata. Among them, the block header includes the block number, the hash value of the current block (hash calculation from all transactions, hash algorithm SHA256), and the hash value of the previous block header. Each block is tight junction with each other through hash values, and has the non processability of anti-collision brought by the hash algorithm. On the other hand, Fabric uses Membership Service Provider (MSP) based on a Public Key Infrastructure (PKI) system to create and manage a set of X.509 certificates [56] and private keys for members, and writes them to the Genesis block to authenticate the members' authority [57], thereby preventing attackers from attacking Fabric by replacing certificates and private keys. Meanwhile, the complete firefighting IoT data is stored on the IPFS server, which generates the IPFS hash of the firefighting IoT data. Due to the irreversibility of hash operations, the use of IPFS to store firefighting IoT data enhances the scheme's tamper resistance.

Furthermore, we also use AES algorithm, RSA algorithm, Merkle DAG, and data backup to ensure the confidentiality, integrity, and verifiability of firefighting IoT data. Firstly, the firefighting IoT data is encrypted using AES keys, which are securely distributed using RSA algorithms to ensure the confidentiality of data. Then, users use the administrator's RSA public key to encrypt the IPFS hash of firefighting IoT data, then attackers cannot directly obtain the IPFS hash, thus unable to obtain firefighting IoT data. In special circumstances, if attackers eavesdrop on IPFS hashes, they need to further obtain the AES key randomly generated by the user. Due to the use of the administrator's RSA public key for encryption, attackers are unable to access the administrator's RSA private key, making it difficult to decrypt firefighting IoT data, thereby ensuring data security. In addition, the Merkle DAG of IPFS is used to generate IPFS hashes of encrypted firefighting IoT data, ensuring integrity and verifiability. Meanwhile, both IPFS and the machines used by administrators store data simultaneously. When IPFS is attacked, the data on the administrator's machine is still available, reducing the probability of data loss.

C. SYSTEM TESTING AND PERFORMANCE ANALYSIS

1) FIREFIGHTING IoT DATA UPLINKING TO BLOCKCHAIN TEST ANALYSIS

In this section, we conduct the process analysis of uploading the firefighting IoT data to the blockchain, which is divided into the following steps.

(1) The administrator and the user register and obtain a key pair, as shown in Figure 5. Users can get their own key pairs by clicking the Menu button, New Key Pair button and Generate button in turn on the main page. Accordingly, the web server will call the ursa library to generate the RSA algorithm key pair and return it to the web page.

(2) The administrator provides his public key to the user for receiving the file, as shown in Figure 6. In the upload firefighting IoT data page, the user selects the upload file, fills in his private key and the administrator's public key, and clicks the upload button to complete the file upload.

(3) As shown in Figure 7, if the file is uploaded successfully, it indicates that the file has been stored in the IPFS network and the hash of the Fabric transaction is generated, which can be queried in the blockchain browser. The administrator uses its private key to download firefighting IoT data.

(4) The user uses the blockchain browser to query the transaction, as shown in Figure 8. The message is an encrypted IPFS hash, which is generated by encrypting the IPFS hash with the administrator's public key. Therefore, only the administrator can use his private key to decrypt the message, and then download the firefighting IoT data, thus ensuring the secure transmission of data.

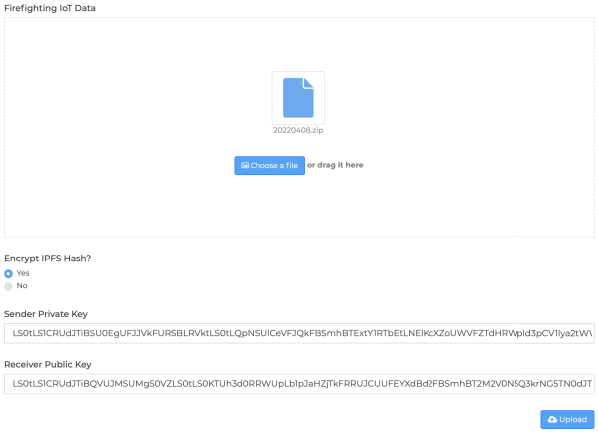

FIGURE 6. The user uploads firefighting IoT data.

FIGURE 7. The uploading results of firefighting IoT data.

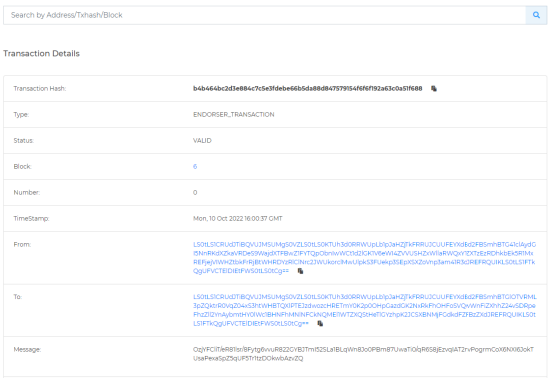

FIGURE 8. Transaction details in the blockchain explorer.

2) SYSTEM STORAGE SPACE USAGE TEST ANALYSIS

The traditional blockchain storage system puts the data into the ledger of each node, while this system stores the firefighting IoT data in IPFS, and the transmission records of the data are stored in the LevelDB database of the peer node of the Fabric network. This system registers users by generating RSA key pairs, and the key pairs are generated using the RSA algorithm. We test the space usage of 1-10 users using the system respectively. For the test data, this system uses one mp4 file (one minute of video surveillance data from a single 768kbps video monitor, which takes up 6.224MB of space) and one JSON file (which takes up about 1KB of space) packaged in a zip file named by a timestamp. The user sends the firefighting IoT data to the administrator account by using the web page, and the data is stored in the backend on the IPFS server in the data center. Optionally, the administrator

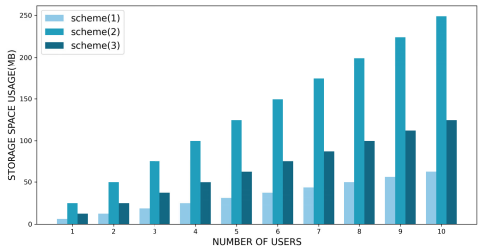

FIGURE 9. Comparison of occupied space under different schemes.

can download the firefighting IoT data from the user using his private key upon receipt.

This experiment tests the space occupation of three schemes: (1) firefighting IoT data is not on the blockchain and only stored locally; (2) all firefighting IoT data is on the blockchain (traditional blockchain storage system); (3) the combination of blockchain and IPFS proposed in this paper is used for firefighting IoT data storage. In the case of 4 peer nodes, a different number of users use different schemes to upload one minute of firefighting IoT data respectively. The result of space occupation of firefighting IoT data for storage is shown in Figure 9. In scheme (1), when the firefighting IoT data is not on the blockchain, the space occupation is the least, but there is a possibility that the data is destroyed and the security of the data cannot be guaranteed; in scheme (2), each peer node stores the backup of firefighting IoT data, which can securely store the firefighting IoT data, but when the number of peer nodes and users increases (ensure security), with the accumulation of time, the occupied space is much larger than scheme (1) and scheme (3); in particular, scheme (3) stores the data in IPFS server, and Fabric only records the transmission records of all firefighting IoT data, compared with scheme (2), on the premise of ensuring the secure storage of firefighting IoT data, the storage space of nodes is saved by about 50% and the storage space occupation is only doubled compared with scheme (1), but the security of firefighting IoT data is greatly improved. Therefore, scheme (3) proposed in this paper improves the security than the traditional centralized storage, and at the same time, saves the node storage space than the common blockchain storage system.

3) SYSTEM THROUGHPUT AND LATENCY TEST ANALYSIS

Fabric 1.4.4 has three built-in consensus algorithms, Solo, Kafka, and Raft. Different consensus algorithms use different system resources [58], which can affect the throughput of contract functions in Fabric, and the security of different consensus algorithms also differs. In order to choose a suitable consensus algorithm, we first test the performance of the Fabric network with the built-in consensus algorithm and the PBFT consensus algorithm respectively, and then analyze the performance test results and the security of the consensus algorithm together, and finally select the PBFT consensus algorithm.

In order to verify the performance of Fabric networks based on different consensus algorithms, the Transactions Per

Second (TPS) [59] in this paper. The testing tool is Tape from the Hyperledger Technical Community Group China, which is used by first cloning the Tape repository and compiling the Tape binaries locally, and then modifying the configuration file and testing the Fabric network according to the documentation. In the test, the Tape is used to stress test the query function and sendhash function of the Fabric network chaincode, and the number of concurrency is set to 10, 20, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000. Each test is conducted 10 times at different levels of concurrency, with an average result expressed as a 95% confidence interval.

In Figures 10 and 11, the three consensus algorithms built in Fabric perform better under different concurrency numbers, with throughput ranging from 21.16 to 768.43 TPS. The Solo consensus algorithm contains only a single sorting node, which does not involve communication between consensus nodes, and the algorithm has a higher TPS, but with only one node, resulting in poorer security. The system throughput level of Kafka and Raft consensus algorithms is equivalent, but the additional management overhead of Kafka cluster is large, and users generally choose Raft consensus algorithm. The PBFT consensus algorithm has the lowest throughput at only 21.16 to 299.38 TPS. The reason is that the three-stage consensus of the PBFT consensus algorithm increases the network communication overhead [60], but ensures the security of blockchain transactions. Although the built-in consensus algorithms have better throughput performance, they have the problems of poor security and high overhead, and they do not support Byzantine fault tolerance, which leads to the security risks of Fabric. The PBFT consensus algorithm exactly makes up for the security defects of the Fabric 1.4.4 network. Therefore, this system uses the PBFT consensus algorithm as the sorting service algorithm of Fabric.

In addition, for all consensus algorithms, the system throughput of the query function and sendhash function under different concurrent numbers is maintained between 21.16 and 768.43 TPS, and the throughput of the query function is higher than that of the sendhash function as a whole. The reason is that the query function only reads the world state, while the sendhash function belongs to the invoke operation, which requires more time to communicate with the orderer node. Although the throughput of sendhash function and query function under PBFT is 21.16 TPS at the lowest, compared with the Bitcoin throughput of 7 TPS and the Ethereum throughput of 15 TPS [61], the throughput has increased by 202% and 41%, respectively. Furthermore, we also test the latency of these functions. We use the Caliper to send 2000 transactions to the query and sendash functions, with an average delay of 0.55s and 0.75s, respectively. This means that the average transaction time of the Fabric network using the PBFT consensus algorithm is completed within 1 second. Correspondingly, the transaction confirmation time of Bitcoin is 10 minutes [62]. Compared with the public blockchain, the transaction confirmation delay of Fabric is extremely low. Therefore, the blockchain of Fabric embedded in PBFT can still better meet the needs of enterprises and ensure the reliability of the blockchain network.

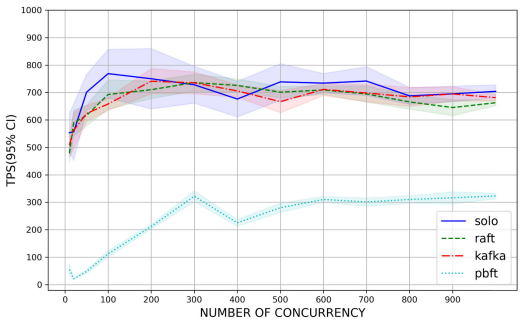

FIGURE 10. Throughput of query function under different concurrency.

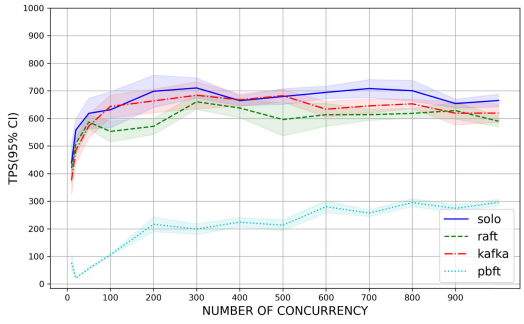

FIGURE 11. Throughput of sendhash function under different concurrency.

4) ENCRYPTION ALGORITHM PERFORMANCE TEST ANALYSIS

In our scheme, we use AES and RSA algorithms to protect firefighting IoT data, IPFS Hash, and the user's information, respectively. In this case, the AES algorithm runs on the department user's PC and RSA algorithm runs on the server. The firefighting IoT devices in this scheme are only used to collect firefighting IoT data and no encryption algorithm is involved. In more details, we use AES symmetry key to encrypt the firefighting IoT data, RSA public key to encrypt IPFS Hash, and RSA 1024 bits to generate user's account. We test the above applications involving encryption algorithms, and each is tested ten times to take the average. The average time required to encrypt and decrypt simulated firefighting IoT data using AES is 24.98ms and 24.78ms, respectively; the average time required to encrypt and decrypt IPFS Hash using RSA is 57.50μs and 417.12μs; the average time required to generate user accounts, i.e., public-private key pairs, using RSA is 21.22ms. From the above data, it can be seen that the encryption (decryption) time of the encryption algorithms in our scheme can be basically ignored, which can ensure the safe transmission of firefighting IoT data with low time overhead.

VII. CONCLUSION

In this paper, we design and implement a distributed storage framework for firefighting IoT data based on blockchain and IPFS. In this framework, a large amount of firefighting IoT data is stored off-chain using IPFS, and the blockchain only stores the transmission records of firefighting IoT data, which saves the hard disk storage space of each blockchain node and ensures the security and low-cost of data storage. Meanwhile, RSA and AES algorithms are used to encrypt firefighting IoT data and transmission records, which improves the security of data storage and transmission. By adding the PBFT consensus algorithm to Fabric, the blockchain network supports Byzantine fault tolerance and enhances the reliability of the blockchain system. In addition, through the blockchain browser, users can query the transmission records of all firefighting IoT data and realize real-time supervision of the transactions. Simulation experimental results and security analysis show that the proposed firefighting IoT data storage system can store firefighting IoT data securely and at low cost, and ensure the integrity, validity and non-tamperability of firefighting IoT data. Compared with the traditional blockchain storage system, it saves storage space with high system throughput and low latency overhead. In the future work, we will consider allowing more organizations to join the blockchain, including government and firefighting departments, and use access control to achieve the data sharing security, while adding privacy protection functions, so as to further improve the integrity and practicability of the framework.

ACKNOWLEDGMENT

(Dezheng Jin is co-first author.)

REFERENCES

[1] Z. Guowei, Y. Su, Z. Guoqing, F. Pengyue, and J. Boyan, "Smart firefighting construction in China: Status, problems, and reflections," *Fire Mater.*, vol. 44, no. 4, pp. 479–486, Jun. 2020.

[2] A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, and A. A. Khan, "A review and state of art of Internet of Things (IoT)," *Arch. Comput. Methods Eng.*, vol. 29, pp. 1395–1413, Jul. 2021.

[3] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th Int. Joint Conf. INC, IMS IDC*, Aug. 2009, pp. 44–51.

[4] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4481–4493, Aug. 2016.

[5] A. J. Schmitt, S. A. Sun, L. V. Snyder, and Z.-J.-M. Shen, "Centralization versus decentralization: Risk pooling, risk diversification, and supply chain disruptions," *Omega*, vol. 52, pp. 201–212, Apr. 2015.

[6] A. A. Khan, M. A. Khan, K. Leung, X. Huang, M. Luo, and A. Usmani, "A review of critical fire event library for buildings and safety framework for smart firefighting," *Int. J. Disaster Risk Reduction*, vol. 83, Dec. 2022, Art. no. 103412.

[7] X. Wu, X. Zhang, Y. Jiang, X. Huang, G. G. Q. Huang, and A. Usmani, "An intelligent tunnel firefighting system and small-scale demonstration," *Tunnelling Underground Space Technol.*, vol. 120, Feb. 2022, Art. no. 104301.

[8] N. Khan, D. Lee, C. Baek, and C.-S. Park, "Converging technologies for safety planning and inspection information system of portable firefighting equipment," *IEEE Access*, vol. 8, pp. 211173–211188, 2020.

[9] S. S. Ramasamy, N. Suyaroj, and N. Chakpitak, "Forest protection by fire detection, alarming, messaging through IoT, blockchain, and digital technologies in Thailand Chiang Mai forest range," in *Data Science and Security*. Cham, Switzerland: Springer, 2022, pp. 167–179.

[10] R. Gürfidan and M. Ersoy, "A new approach with blockchain based for safe communication in IoT ecosystem," *J. Data, Inf. Manage.*, vol. 4, no. 1, pp. 49–56, Mar. 2022.

[11] J. Chi, Y. Li, J. Huang, J. Liu, Y. Jin, C. Chen, and T. Qiu, "A secure and efficient data sharing scheme based on blockchain in Industrial Internet of Things," *J. Netw. Comput. Appl.*, vol. 167, Oct. 2020, Art. no. 102710.

[12] Y. Liu and S. Zhang, "Information security and storage of Internet of Things based on block chains," *Future Gener. Comput. Syst.*, vol. 106, pp. 296–303, May 2020.

[13] Y. Li, Y. Tu, J. Lu, and Y. Wang, "A security transmission and storage solution about sensing image for blockchain in the Internet of Things," *Sensors*, vol. 20, no. 3, p. 916, Feb. 2020.

[14] X. Wang, C. Wang, K. Zhou, and H. Cheng, "ESS: An efficient storage scheme for improving the scalability of Bitcoin network," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 1191–1202, Jun. 2022.

[15] S. Zhi-Xin, Z. Xin, X. Feng, and C. Lu, "Survey of storage scalability on blockchain," *J. Softw.*, vol. 32, no. 1, pp. 1–20, 2021.

[16] T. Wu, P. L. Yeoh, G. Jourjon, and K. Thilakarathna, "Mapchain: A DHT-based dual-blockchain data structure for large-scale IoT systems," in *Proc. IEEE 7th World Forum on Internet Things (WF-IoT)*, Jun./Jul. 2021, pp. 177–182.

[17] S. Ali, G. Wang, B. White, and R. L. Cottrell, "A blockchain-based decentralized data storage and access framework for PingER," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 1303–1308.

[18] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep. 2019.

[19] S. S. Hasan, N. H. Sultan, and F. A. Barbhuiya, "Cloud data provenance using IPFS and blockchain technology," in *Proc. 7th Int. Workshop Secur. Cloud Comput.*, Jul. 2019, pp. 5–12.

[20] Q. Zheng, Y. Li, P. Chen, and X. Dong, "An innovative IPFS-based storage model for blockchain," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Dec. 2018, pp. 704–708.

[21] R. Norvill, B. B. Fiz Pontiveros, R. State, and A. Cullen, "IPFS for reduction of chain size in Ethereum," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1121–1128.

[22] Z. Ullah, B. Raza, H. Shah, S. Khan, and A. Waheed, "Towards blockchain-based secure storage and trusted data sharing scheme for IoT environment," *IEEE Access*, vol. 10, pp. 36978–36994, 2022.

[23] P. A. Lobo and V. Sarasvathi, "Distributed file storage model using IPFS and blockchain," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*, Oct. 2021, pp. 1–6.

[24] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[25] S. Ben Toumia, C. Berger, and H. P. Reiser, "An evaluation of blockchain application requirements and their satisfaction in Hyperledger Fabric: A practical experience report," in *Proc. IFIP Int. Conf. Distrib. Appl. Interoperable Syst.* Cham, Switzerland: Springer, 2022, pp. 3–20.

[26] H. Kuzuno and C. Karam, "Blockchain explorer: An analytical process and investigation environment for Bitcoin," in *Proc. APWG Symp. Electron. Crime Res. (eCrime)*, Apr. 2017, pp. 9–16.

[27] Y. Hassanzadeh-Nazarabadi, A. Küpçü, and Ö. Özkasap, "LightChain: Scalable DHT-based blockchain," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 10, pp. 2582–2593, Oct. 2021.

[28] Q. Xinyi, C. Yuling, Z. Zhengqiang, T. Yuanchao, and L. Tao, "Blockchain storage expansion model based on Chinese remainder theorem," *J. Comput. Appl.*, vol. 41, no. 7, p. 1977, 2021.

[29] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[30] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in IPFS," *IEEE Access*, vol. 8, pp. 59389–59401, 2020.

[31] R. Kumar, R. Tripathi, N. Marchang, G. Srivastava, T. R. Gadekallu, and N. N. Xiong, "A secured distributed detection system based on IPFS and blockchain for industrial image and video data security," *J. Parallel Distrib. Comput.*, vol. 152, pp. 128–143, Jun. 2021.

[32] K. Azbeg, O. Ouchetto, and S. Jai Andaloussi, "BlockMedCare: A healthcare system based on IoT, blockchain and IPFS for data management security," *Egyptian Informat. J.*, vol. 23, no. 2, pp. 329–343, Jul. 2022.
[33] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[34] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second," *Int. J. Netw. Manage.*, vol. 30, no. 5, pp. 455–463, Sep. 2020.
[35] J. Sousa, A. Bessani, and M. Vukolic, "A Byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 51–58.
[36] H. Sun, W. Zhang, X. Wang, Z. Ma, L. Huang, and X. Li, "A robust Byzantine fault-tolerant consensus algorithm against adaptive attack based on ring signature and threshold signature," *Acta Automatica Sinica*, vol. 49, no. 7, pp. 1471–1482, 2021.
[37] S. Duan and H. Zhang, "Foundations of dynamic BFT," Cryptol. ePrint Arch., White Paper 2022/597, 2022. [Online]. Available: https://eprint.iacr.org/2022/597
[38] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, vol. 107, pp. 841–853, Jun. 2020.
[39] S. M. H. Bamakan, A. Motavali, and A. B. Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Syst. Appl.*, vol. 154, Sep. 2020, Art. no. 113385.
[40] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun.*, vol. 11, nos. 1–2, pp. 51–64, 2018.
[41] N. Szabo, "Smart contracts: Building blocks for digital markets," *EXTROPY, J. Transhumanist Thought 16*, vol. 18, no. 2, p. 28, 1996.
[42] L. Li, T. Zhang, G. Sun, D. Jin, and N. Li, "A fair, verifiable and privacy-protecting data outsourcing transaction scheme based on smart contracts," *IEEE Access*, vol. 10, pp. 106873–106885, 2022.
[43] J. Benet, "IPFS—Content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.
[44] S. Sarmady, "A survey on peer-to-peer and DHT," 2010, *arXiv:1006.4708*.
[45] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. Workshop Econ. Peer-to-Peer Syst.*, vol. 6, 2003, pp. 68–72.
[46] D. Spinellis, "Git," *IEEE Softw.*, vol. 29, no. 3, pp. 100–101, May 2012.
[47] D. D. F. Mazières, "Self-certifying file system," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2000.
[48] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, pp. 1–15.
[49] M. Antwi, A. Adnane, F. Ahmad, R. Hussain, M. H. U. Rehman, and C. A. Kerrache, "The case of HyperLedger Fabric as a blockchain solution for healthcare applications," *Blockchain, Res. Appl.*, vol. 2, no. 1, Mar. 2021, Art. no. 100012.
[50] T. Hewa, M. Ylianttila, and M. Liyanage, "Survey on blockchain based smart contracts: Applications, opportunities and challenges," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102857.
[51] P. Zhang and J. Song, "Research advance on efficiency optimization of blockchain consensus algorithms," *Comput. Sci.*, vol. 47, no. 12, pp. 296–303, 2020.
[52] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf. (Usenix ATC)*, 2014, pp. 305–319.
[53] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proc. USENIX Int. Workshop Netw. Meets Databases*, vol. 11, 2011, pp. 1–7.
[54] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173–186.
[55] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to Docker and analysis of its performance," *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.
[56] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X. 509 Internet public key infrastructure online certificate status protocol-OCSP," IETF Netw. Work. Group, San Diego, CA, USA, Tech. Rep. 2560, 1999.
[57] D. Ravi, S. Ramachandran, R. Vignesh, V. R. Falmari, and M. Brindha, "Privacy preserving transparent supply chain management through Hyperledger Fabric," *Blockchain, Res. Appl.*, vol. 3, no. 2, Jun. 2022, Art. no. 100072.
[58] H. Samy, A. Tammam, A. Fahmy, and B. Hasan, "Enhancing the performance of the blockchain consensus algorithm using multithreading technology," *Ain Shams Eng. J.*, vol. 12, no. 3, pp. 2709–2716, Sep. 2021.
[59] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of Hyperledger Fabric," in *Proc. Crypto Valley Conf. Blockchain Technol. (CVCBT)*, Jun. 2018, pp. 65–74.
[60] L. Zhang, H. Xu, O. Onireti, M. A. Imran, and B. Cao, "How much communication resource is needed to run a wireless blockchain network?" *IEEE Netw.*, vol. 36, no. 1, pp. 128–135, Jan. 2022.
[61] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain, Res. Appl.*, vol. 3, no. 2, Jun. 2022, Art. no. 100067.
[62] T. Pflanzner, H. Baniata, and A. Kertesz, "Latency analysis of blockchain-based SSI applications," *Future Internet*, vol. 14, no. 10, p. 282, Sep. 2022.

LINA LI received the M.S. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, in 2006, and the Ph.D. degree in computer system architecture from the College of Computer Science and Technology (CCST), Jilin University, Changchun, in 2019. She is currently an Associate Professor with the College of Computer Science and Technology, Changchun University. Her research interests include blockchain, deep learning, and cloud computing.

DEZHENG JIN was born in Anyang, China, in 1997. Currently, he is a Postgraduate Student with Changchun University. He has a strong interest in the direction of blockchain and has conducted some research about smart contract and consensus mechanism in this field. He has participated in a provincial scientific research project based on blockchain technology. He has published an EI conference paper and applied for an invention patent. His main research interests include blockchain and deep learning.

TINGTING ZHANG was born in Changchun, Jilin, China, in 1997. She is currently pursuing the master's degree with the College of Cyber Security, Changchun University, Jilin. She has participated in several provincial scientific research projects with the College of Computer Science and Technology, and relevant research results have applied for an invention patent and published a SCI journal article. Her research interests are blockchain and smart contract.

NIANFENG LI received the Ph.D. degree in mechatronics engineering from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences. He is currently a Professor and the Dean of the College of Computer Science and Technology (CCST), Changchun University, China, where he is also a Vice Director of the Biomedical Engineering Research and Development Center. His current major research interests include image processing, somatosensory technology, rehabilitation training systems, and campus safety technology.