Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.017516

ARTICLE

FileWallet: A File Management System Based on IPFS and Hyperledger Fabric

Jienan Chen, Chuang Zhang, Yu Yan and Yuan Liu*

Software College of Northeastern University, Shenyang, China

*Corresponding Author: Yuan Liu. Email: liuyuan@swc.neu.edu.cn

Received: 16 May 2021 Accepted: 25 August 2021

ABSTRACT

Online file management systems enable cooperatively editing and sharing. However, due to the cost of communication and storage infrastructures, traditional online file management services, e.g., Google Drive and OneDrive, usually provide limited storage space and relatively low download speed for free users. To achieve better performance, ordinary users have to purchase their expensive services. Moreover, these file management systems are based on centralized architecture and bear the privacy leakage risk, because users' personal files are stored and controlled by their servers. To address the above problems, we propose a peer-to-peer (P2P) file management system based on IPFS and Hyperledger Fabric, named as FileWallet, which can serve as a personal wallet for individual users or organizations to store and share their files in a secure manner. In FileWallet, the users form a P2P network and a Fabric network, where P2P network builds the connections and distributed storage network and the Fabric network sustains consistent blockchain ledgers to record file operation related transactions. In our FileWallet, the storage and communication costs are mitigated in the decentralized design, and the file owner can fully control the access permission of the file to preserve the file privacy. The design of the system architecture, main functionalities, and system implementations are presented in this paper. The performance of the system is evaluated through experiments, and the experimental results show its wide applicability and scalability.

KEYWORDS

Blockchain; file sharing; IPFS

Nomenclature

Un	Username
Up	User profile
K	Directory key
Kp	Parent directory key
Dm	Directory metadata
Dn	Directory name
Dv	Directory visibility
Fm	File metadata
Fn	File name

1 Introduction

A file management system is a program that is used for file maintenance in a computer system [1]. For example, Windows Explorer is the default file management system for Windows and Finder is for macOS. However, these file management systems are designed for managing local files. With the popularization of the internet, the need for file sharing and cooperatively editing impels the development of online file management systems, e.g., Google Drive [2]. With the exponentially increasing file based data and documents, it becomes challenging for an online file management system to ensure the efficient accessibility with the file content privacy preserved [3]. In other words, the files should be able to be swiftly accessed by only the permissioned or authenticated users. The existing systems bear the following two shortcomings:

- Privacy leakage: Since all the data files are hosted in a centralized server, users cannot directly control how their files are maintained and stored. From the technical perspective, any malicious maintainer is able to leak users' data without authentication if the users upload their files in plain-text format. The data leakage events often happen in recent years, e.g., half a billion Facebook users' information was leaked by an insider attacker in 2019 and posted on hacking website for sale in April 2021 [4].
- High cost: Many online file management systems like Google Drive, Dropbox, and OneDrive use the centralized architecture, and these systems are managed by IT companies. In order to provide an online storage service, these companies have to pay the maintenance costs such as electricity consumption, device renewal fee and network bandwidth fee. For individual users, only limited free storage space is offered at a low download speed. Users having a larger storage requirement have to purchase the service membership at a high price, e.g., $15 per month, which increases the financial burden of users.

In order to overcome the above two shortcomings, the design of decentralized file management system is developed and becomes a promising research direction. IPFS [5], which is the abbreviation of inter planetary file system, is an innovative distributed file storage system that is powered by peer-to-peer networks. It has high security and integrity properties benefiting from the content addressing and distributed hash table (DHT) techniques. The peers in the IPFS network can exchange files effectively without trusting each other. Meanwhile, Hyperledger Fabric [6] is an open-source permissioned blockchain platform started by Linux Foundation in 2015. It has many advantages like high security, high availability, and low confirmation latency. Based on IPFS and Fabric, this study aims to propose a decentralized file management system.

Although vanilla IPFS is powerful enough for many scenarios, we still discover some challenging flaws that have negative impacts on user experiences.

- Poor support of file content updation: The content identifier (CID) of a file is often immutable, which is generated based on hashing file content. However, a file may require to be updated by its owner, which results in the change of the file's CID. IPFS proposes inter planetary name system (IPNS) as the expedient measure for addressing this issue. But it is terribly slow while updating or accessing the files if there are few connected peers.
- No cross-device synchronization: The CID is a long string that is hard for human beings to remember. When a user wants to access the newly added file on a new device, the user must use some storage media like a textbook to record the CID of files or directories. So, it is very inconvenient for the user who has multiple devices.

In this paper, we propose FileWallet, an online file management system that takes advantage of IPFS and Hyperledger Fabric to overcome the existing shortcomings and challenges. In

FileWallet, files are stored and managed by IPFS. In order to support file updation, we store the information of the directory that has many file CIDs in the Hyperledger Fabric ledger. The ledger is a key-value database, and we can use the key to access the CID of a file or a directory. If a user wants to update a file in the directory, the user can propose a transaction to amend the directory information by using the key. Therefore, others can always access the latest files with a constant directory key. Meanwhile, In order to enable a user to access its files across multiple devices, we use the hash of its certificate as the key of the user profile that contains the user's basic information, like the user's root directory key, where the certificate is managed by membership service providers (MSP) in Fabric. So, the user can access his user profile on different devices as long as the user owns the same certificate. When a user accesses a file, the user needs to search for the content provider by using the file CID in the IPFS network and establish a P2P connection with the content provider for the purpose of file transmission. In addition to MSP, Hyperledger Fabric has a component called "channel" that is used for secure communication between different organizations, and the data of different channels are isolated. So, users can prevent file leakage by only establishing channels with the organizations they trust. Taking the advantages of decentralized architecture of IPFS and Hyperledger Fabric, FileWallet is also a decentralized system that does not bear the centric server cost. Its download speed depends on the upload speed of the content providers and the number of the providers. The download speed will be higher if there are more replicas in the IPFS network where the file is stored on the uploader's device initially and distributed to other nodes when others query the file. So, users do not have to pay for extra storage capacity. The main contributions of this study are summarized as follows:

- A P2P file management system architecture is proposed based on IPFS and Hyperledger Fabric, which can effectively mitigate the privacy leakage and high cost issues.
- A Fabric based file maintenance solution is designed for IPFS to support file content updation with the same CID.
- A cross-device client is implemented to enable file synchronization among multiple devices, which also support version control, cooperation, subscription, and data security.

2 Related Work and Preliminaries

In the field of computer science, many file/directory management systems have been proposed to manage local documents and support operation system operations [7,8]. With the widespread of Internet based applications, the files are often exchanged and shared between different entities for the purpose of cooperation or information propagation, and online file management become dominant. Furthermore, The need for good user experience and high efficiency encourages people to investigate online distributed file management systems [9], for example BitTorrent [10]. The existing studies in file management focus on improving the search, read, and write efficiency [11,12]. However, they neglect the file privacy and security issues [13].

The blockchain technology, born in Bitcoin [14], is potential to construct trust relationships among strangers in a decentralized manner, which is potential to secure file privacy and security [15]. There are many studies applying IPFS to overcome the storage cost of the growing size of the blockchain ledger. The authors in [16] proposed a new scheme that replaces the block body data with the transaction hash generated by IPFS. Similarly [17] issued a measure that uses the CID of the transaction to replace the raw data in the block body. However, the purpose of these papers is to reduce the storage usage of miners. But the idea that uses CID of data in transactions inspired us to conceive FileWallet.

Meanwhile, many blockchain based file management systems have been proposed in the literature based on IPFS and Hyperledger Fabric [18]. BlockIPFS [19] proposes a new approach to implement the traceability of file access through Hyperledger Fabric. In general, BlockIPFS stores the metadata of a file in the ledger and each time anyone who want to alter or access the file metadata have to submit a transaction to record their operations. However, the access authentication cannot maintain when a file is updated. In other words, the user may do not know the latest version of the file unless the others inform them. Filecoin [20] is a famous application of IPFS, where FileCoin is a cryptocurrency to incentivize peers to store files. Users have to pay digital currency if they want to store some data in Filecoin. Therefore, if a user wants to put data into Filecoin frequently, the cost will be tremendously high.

2.1 Preliminary of IPFS

IPFS [5] enables users to store and transport files in a secure and effective way without a central server. IPFS intrinsically is a content addressing system, which means that IPFS uses Content Identifier (CID) based on file hash to identify and lookup the corresponding file. Unlike other online file management systems that users will send files to the centric server, IPFS does not send files to other peers spontaneously and only disseminates files when other users request the files. When a user adds a file to the IPFS network, IPFS will split a single file into 256 KB chunks. IPFS then build the Merkle DAG of the file and return the root's CID as the file's CID. Finally, IPFS will add a file record that contains the information about the device that possesses the file to the IPFS peers that close to the CID according to the XOR distance. IPFS uses the file's CID and uses Kademlia algorithm for its routing system. Specifically, the address of CID and peer id has the same length, and both of them are a 256-bit number. The peer can know where access the file data by querying the peer that has the same id with the CID. Consequently, the network's peers can obtain file data without a central server and effectively avoid the privacy leakage risk brought by the central server.

2.2 Preliminary of Hyperledger Fabric

Unlike the permissionless blockchain frameworks, e.g., Ethereum and Bitcoin, which are based on a probabilistic consensus algorithm, Hyerledger Fabric uses Practical Byzantine Fault Tolerance (PBFT) as the default consensus algorithm. Peers in the Hyperledger Fabric network are not necessary to calculate complex mathematics puzzles. Hence, Hyperledger Fabric has relatively low power consumption and high efficiency. Furthermore, Hyperledger Fabric also supports customized consensus. This feature enables developers to construct their blockchain network according to their actual requirements. The peers in the Hyperledger Fabric network are grouped into different organizations. An organization can represent an entity in the real world, such as an individual user or an independent department. All the blockchain operation is based on the organization rather than an individual peer.

When a client updates the blockchain ledger of a channel, the client follows three phases as shown in Fig. 1.

Phase 1: The client submits a transaction proposal to the endorsement peers (1.1 in Fig. 1). The endorsement peers validate this proposal and return a signed endorsement (1.2 in Fig. 1). After collecting enough endorsements taking a proportion of 2/3, the client will enter Phase 2.

Phase 2: The client submits a transaction that includes the endorsements to the ordering service in the channel (2.1). Finally, the order service will package the transaction and send it to all peers in the channel (2.2).

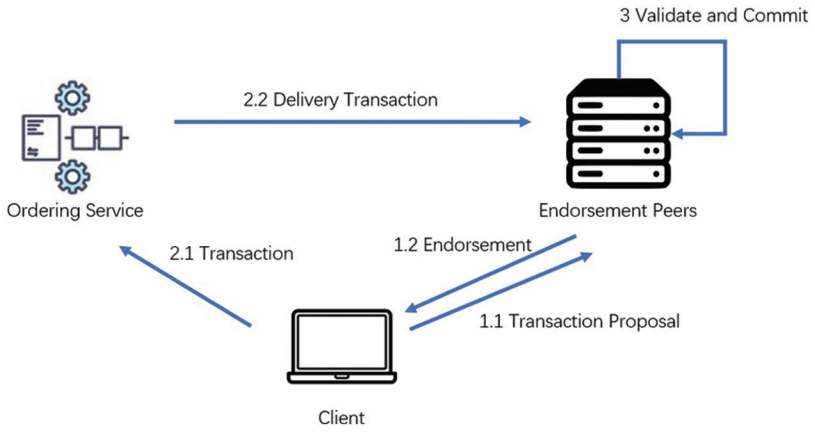

Figure 1: The PBFT consensus phases in hyperledger fabric

Phase 3: When peers receive the transaction, peers will validate the transactions (3). The update will be applied if the transaction passes the validation.

3 FileWallet

In this section, we first introduce the system architecture of the proposed FileWallet system, which is followed by the data structure design. Secondly, we design the smart contracts to achieve the file storage and access functionalities. The main functional models are also provided. Finally, the security of the system is analyzed.

3.1 Architecture

Fig. 2 illustrates the architecture of our system. FileWallet consists of two networks which work cooperatively, namely Hyperledger Fabric network and IPFS network. The Hyperledger Fabric network is responsible for issuing membership certificates. Without a valid certificate, no peer or client is able to interact with the Hyperledger Fabric network. In a production environment, there is usually a TLS CA for issuing TLS certificates when establishing TLS connections. With the TLS connection, the system can effectively avoid the man-in-the-middle attack.

In Hyperledger Fabric network, there are many organizations and each organization further contains a certain number of peers. The peers maintain ledger consensus and execute chaincode or smart contracts, thus a client must interact with the peers so as to access or update the ledger. Another important part of the Hyperledger Fabric network is the channel which is an essential component for communication between different organizations. The network can contain several organizations and an ordering service. The data between channels is completely isolated. Hence, a peer may have multiple ledgers if its organization joins different channels. The ordering service is responsible for ordering and packaging transactions. After a client collects enough endorsements, the client can submit the transaction to the ordering service to update the ledger. Then the ordering service will eventually distribute the transaction block to all the peers in the channel.

In IPFS network, each FileWallet client consists of a user interface, a Hyperledger Fabric client, and an IPFS instance. The Hyperledger Fabric client is used to connect peers and the ordering service in the Hyperldegr Fabric network. To establish the connection with the Hyperledger Fabric network, users have to define a JSON format text file named “Connection Profile” that includes their organization information and the valid user certificate issued by the organization.

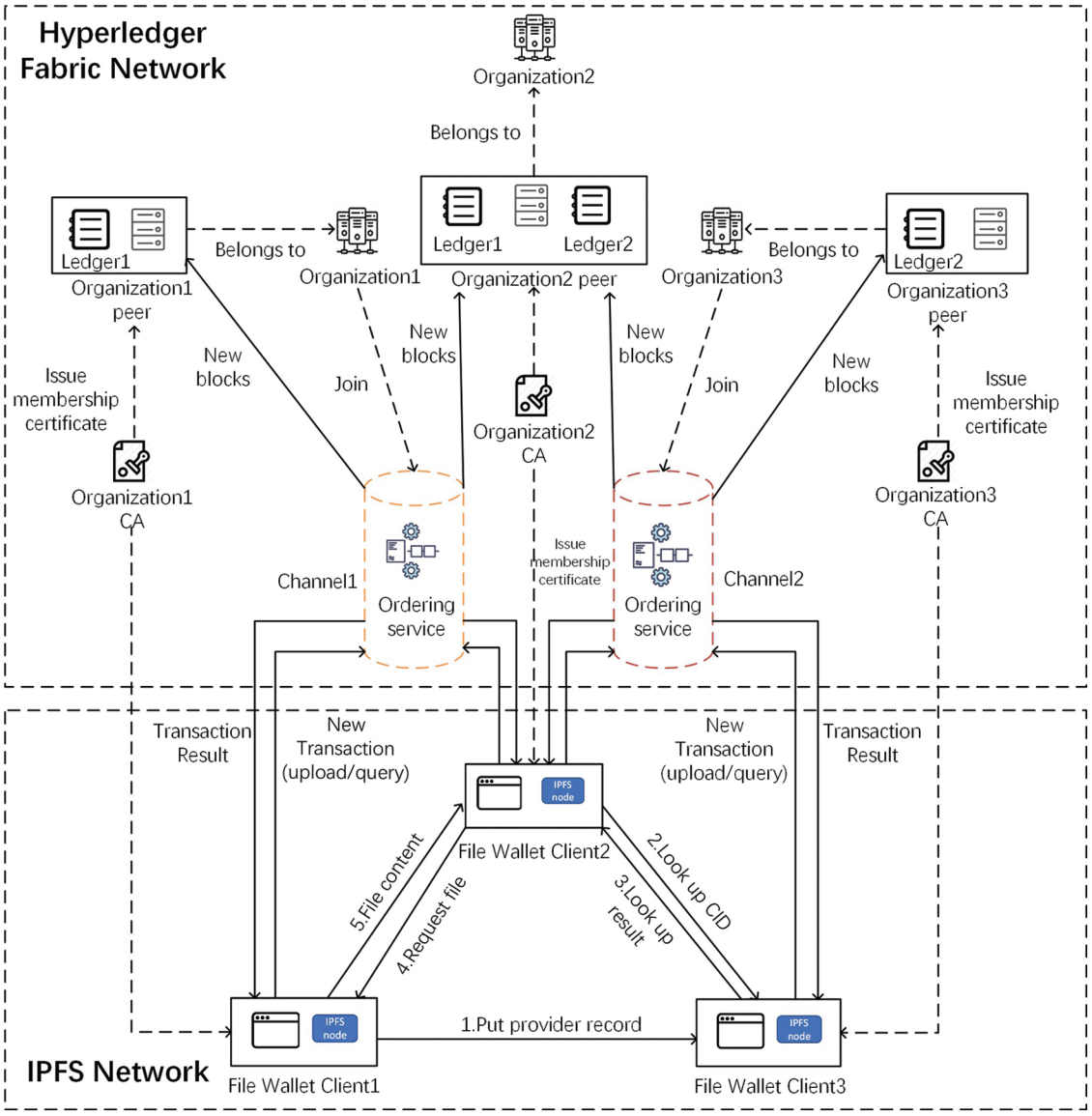

Figure 2: The system architecture

The main system workflow is described as follows. When Client 1 adds a file through the FileWallet client, the client will calculate the CID of and put provider records in several IPFS peers that are close to the file CID according to XOR distance. After that Client 1 can submit a transaction to store the CID into the ledger, and Client 2 can access the CID. The Client 2 can obtain the CID by looking up in the IPFS network using the Kademlia algorithm and know that Client 1 host the file. Finally, Client 2 will establish a connection with Client 1 and get the file.

3.2 Data Structure Design

Hyperledger fabric uses the key-value database to store the ledger state. For the sake of simplicity, we store data in JSON format.

3.2.1 User Profile

User profile records the user's metadata. We use the hash of the user's certificate as the user id and the key of the Up. When the user queries or submits a transaction, the called chaincode verify its client's identity by calculating the certificate's hash. Table 1 shows the data structure of Up.

Table 1: Data structure of user profile

Name	Type	Description
id	string	The id of the user.
name	string	The username of the user.
root	array	The key of the user's root directory.

3.2.2 Directory

The directory data structure records the metadata of a specific directory, which is denoted by Dm. Table 2 shows the data structure of the directory metadata.

Table 2: Data structure of directory metadata

Name	Type	Description
name	string	The name of the directory.
directories	array	An array of subdirectory key.
files	array	An array of file metadata.
creator	string	The creator's id of the directory.
date	number	The creation timestamp of the directory.
cooperators	array	The array of cooperator id.
subscribers	array	The array of subscriber id.
visibility	boolean	The visibility of this directory.

The directories and files are two arrays that record the subdirectories and file content of the current directory. The client can access the subdirectories according to the keys in the "directories" array. The "files" is an array of JSON objects that record the specific content of files. The data structure of the file JSON object will be introduced later.

The cooperators of a directory have the full privilege of this directory. So, a cooperator can add or remove the subdirectories and files of the directory. Besides, the directory creator can invite new cooperators or subscribers to this directory. However, the subscribers of the directory can only read the content of the directory and do not have the privilege to update the directory.

The visibility means that whether the directory can be subscribed by any users without the invitation of cooperators. If the cooperators decide to publish the directory to the public, they can simply change the visibility value as true and send the share link to others. Finally, others can subscribe to the directory through the share link.

3.2.3 File Metadata

The file metadata contains the following fields, as shown in Table 3. Normally, it is embedded in the directory metadata.

Table 3: Data structure of file metedata

Name	Type	Description
cid	string	The CID of the file.
addedDate	number	File added timestamp.
updatedDate	number	File updated timestamp.
name	string	The name of the file.

3.3 Smart Contract Design

To achieve the functionalities of our FileWallet, we designed 13 smart contract functions, which are described in Table 4.

Table 4: Smart contract functions

No.	Function name	Input	Output	Description
1	InitiateUserProfile	Un	Up	Initiate Up for a new user.
2	ReadUserProfile	None	Up	Read current user's profile.
3	NewDirectory	Dn, Kp	K	Create a directory in ledger.
4	AddDirectory	Kp, K	Dm	Add a directory.
5	RemoveDirectory	Kp, K	Dm	Remove a directory.
6	ChangeVisibility	K, Dv	Dm	Change the visibility of directory.
7	ReadDirectory	K	Dm	Read directory metadata from ledger.
8	AddFile	K, Fm	Dm	Add files to a directory.
9	RemoveFile	K, file name	Dm	Remove files from a directory.
10	AddCooperator	K, invitee id	Dm	Add a cooperator to a directory.
11	RemoveCooperator	K, cooperator id	Dm	Remove a cooperator from directory.
12	AddSubscriber	K, invitee id	Dm	Add a subscriber to a directory.
13	RemoveSubscriber	K, subscriber id	Dm	Remove a subscriber to a directory.

3.4 Functional Module Design

In this section, we introduce the detailed design of the main functional modules in FileWallet, including directory management, file management, and file sharing.

3.4.1 Directory Management

This functional module is responsible for managing the directory in ledger. Its specific Procedures 1 are shown as follows:

Procedure 1: Directory Management Procedures

```
procedure ReadDirectory(K)
Dm ← getState(K)
if The user has read privilege then
return Dm
else
Throw privilege error
procedure AddDirectory(Kp, K)
Dm ← ReadDirectory(Kp)
if The user has write privilege then
Add key to directories of Dm
putState(K, Dm)
return Dm
else
Throw privilege error
procedure NewDirectory(Kp, Dn):
T ← getT XT imestamp()
id ← getUserID()
K ← SHA256(id + T + Dn).
Dm ← newDirectoryMetadata(N, T, id)
AddDirectory(Kp, Dm)
putState(K, Dm)
return Dm
procedure RemoveDirectory(Kp, K)
Dm ← readDirectory(Kp)
if The user has write privilege then
Remove K from directories of Dm
putState(K, Dm)
return Dm
else
Throw privilege error
procedure ChangeVisibility(K, newVisibility)
Dm ← readDirectory(K)
if The user has write privilege then
set visibility of Dm to newVisibility
putState(K, Dm)
return Dm
else
Throw privilege error
```

3.4.2 File Management

The file management is the essential function of the system. In traditional centralized file management systems, users have to take much time to upload their files because of limited network bandwidth. However, IPFS doesn't have such problem, and the time of adding a file solely depends on local CPU performance and I/O speed of the hard drives. The main Procedures 2 is shown as follows:

Procedure 2: Directory Management Procedures

procedure AddFile(K, Fm)
Dm ← readDirectory(K)
if The user has write privilege and file name has no conflict then
Add Fm to files of DM putState(K, Dm) return Dm
else
Throw privilege error
procedure RemoveFile(K, Fn)
Dm ← readDirectory(K)
if The user has write privilege then
Remove the metadata of the file named Fn from Dm putState(K, Dm)
return Dm
else
Throw privilege error

3.4.3 File Sharing

Data sharing is one of the most highlight features of FileWallet. Because IPFS is using content addressing storage model, the CID of a file is calculated based on the file content. Whenever a user wants to update the file content, the user will generate a new CID. Thus, it will be very troublesome if a file is shared with many users and updated frequently. Currently, IPFS uses IPNS to address the mutability issue of files. IPNS is a feasible solution provided by the IPFS. IPNS can generate a key K that is the hash of the peer public key for the IPNS record. The record includes the hash of a link and a signature signed by the corresponding private key to be stored by those peers whose id close to SHA256(/ipfs/K) in XOR distance. Thus, other users can access the latest content by looking up/ipfs/K. However, this method is inefficient because both the sender and receivers are required to has a connection with the peer that is hosting the IPNS record. It may take long time to update or look up the file in a poor network environment where only a few peers are available. In our system, we propose an alternative solution which is an important component of file sharing module. In FileWallet, we integrate Hyperledger Fabric with IPFS, and we store the file-sharing information in the ledger rather than the IPFS network. Because all peers in a channel have a consistent ledger, the client can access the latest file's CID from one of the peers directly without the time-consuming look-up process.

We design two main functions in file sharing model: cooperation and subscription as shown in Procedure 3. The difference between cooperation and subscription is that cooperators have both write and read privileges and the subscribers only have read privilege. The following procedure shows how the system share files between different users.

Procedure 3: File Management Procedure

procedure AddFile(K, Fm)
Dm ← readDirectory(K)
if The user has write privilege and file name has no conflict then
Add Fm to files of DM
putState(K, Dm)
return Dm
else
Throw privilege error
procedure RemoveFile(K, Fn)
Dm ← readDirectory(K)
if The user has write privilege then
Remove the metadata of the file named Fn from Dm
putState(K, Dm)
return Dm
else
Throw privilege error

3.5 Analysis

This section analyzes the properties of the proposed FileWallet system in the aspects of security, data availability, and traceability.

3.5.1 Security and Privacy

The security issue of our system is mainly affected by the reliability of the Hyperledger Fabric network and the privacy issue is caused by the file content leakage.

Because Hyperledger Fabric is a permissioned blockchain framework, the administrator of an organization can manage the certificates to control the members in the organization. No one can access the organization's data without its permission. Moreover, since the data between channels is isolated, a member of an organization can only access the data of another channel by joining that channel. The administrator of the organization can refuse to join the channel if the administrator does not trust the participants of that channel. As a result, the organization can fully control its data security.

In IPFS, the file provider records in stored in the corresponding peers in the IPFS network. There is a potential data leaking risk because the malicious IPFS peers may access the private data according to provider records, especially in the scenario of a private IPFS network. In FileWallet, the users encrypt their files before adding them to the IPFS network to avoid the privacy leakage risk.

3.5.2 Data Availability

When a file is added in IPFS network, the file is not sent to other peers initially. Others can only access the file when the uploader's device is online at the first time of the file being accessed. If others download the files from the uploader, they will normally keep a replica of the file. As a result, the risk of data lost could be reduced with the number of the replicas. Besides, more replicas can also improve the download speed and reduce access latency.

Futhermore, there is no single point of failure in our system because our system is basically decentralized. If there are enough peers and ordering nodes in the Hyperledger Fabric network, other peers can replace the failed peer because they are symmetrical and take the same responsibilities.

During the development, we discover that the Hyperledger Fabric Network will have multi-version concurrency control (mvcc) errors when updating a key's value concurrently, so we design a retry mechanism on the client. To be specific, the client will retry after several seconds when it gets the mvcc error while interacting with the Hyperledger Fabric network. Hence, the user will not perceive this error in our system.

3.5.3 Traceability

Traceability is one of the fundamental features of a blockchain system, so it is possible for developers to query old versions of a value through its key. Thanks to the powerful Hyperledger Fabric chaincode API, version control can be implemented easily in our system. In FileWallet, the key of the directory will not change despite its content update. Hence, a user can track all the history of a directory as long as the user has read privilege.

4 System Implementation

This section will demonstrate how we implement the system, including the environment setting, network development, and FileWallet client.

4.1 Environmental Settings

First of all, we need to install the necessary dependencies for the development. The following Table 5 shows our development dependencies. A different version of dependencies may cause some unknown errors, so the system may not work in some rare cases like legacy dependencies.

Table 5: The version of dependencies in our implementation environment

Dependencies	Version
Go	1.13.15
Node	14.15.1
Docker Engine	20.10.2
Hyperledger Fabric	2.3
Hyperledger Fabric CA	1.4
IPFS	0.80

4.2 Network Deployment

4.2.1 Certificates Generation

Certificates are essential in the Hyperledger Fabric network. Hyperledger Fabric uses the X.509 standard for all certificates. There are two purposes of using certificates in Hyperledger Fabric network. The first purpose is to prove the valid identity of one specific node. Without a valid certificate, it is impossible for a node to interact with the network. The second purpose is to establish a TLS connection that can prevent a man-in-the-middle attack.

In FileWallet, we use Hyperledger Fabric CA to generate all necessary certificates. The following steps show how we generate certificates.

(1) Pull the latest docker image of Hyperledger Fabric CA from docker hub or download the latest version of binary.
(2) Customize CA configurations by amending the template configuration file.
(3) Run the CA container or binary.
(4) Use the CA client to register and enroll admins, peers, orderers, and clients.

After these operations, the organization administrator distributed these certificates to different entities to allow them joining the network.

4.2.2 Peer Deployment

First, we need to set up organizations. In a production environment, each organization usually contains several peers and one CA that is responsible for issuing membership certificates. Moreover, the ordering service is also critical to the network. After setting up organizations and the ordering service, we need to create a channel for them to communicate with each other. Here is how we set up the peer.

(1) Customize the configuration file for organization peers and the start peers.
(2) Create the ordering service configuration and start the ordering nodes.
(3) Set up the channel configuration and generate the genesis block.
(4) Each organization and the ordering service join the channel by using the genesis block.

Each organization can join different channels to communicate with different organizations by repeating Step 3 and Step 4.

4.2.3 Smart Contract Deployment

After deploying the Hyperledger Fabric network, we install the chaincode before deploying the designed smart contracts. Practically, we use Go as the smart contract language because it has the best official maintenance. The detailed settings of deploying a smart contract are shown below:

(1) Package the smart contract into single compressed file.
(2) Copy the chaincode to peers and install it.
(3) Approve the chaincode for the channel using the organization administrator identity on all organization.
(4) Using one organization to commit the chaincode to the channel.

Finally, the Hyperledger Fabric network is able to response the smart contract calls from clients.

4.3 FileWallet Client

In order to be compatible with various operating systems as much as possible, we implemented the FileWallet Client based on Electron currently. Electron is a prevailing desktop application development framework that enables developers to develop desktop applications with website development techniques like Javascript and CSS. It can also build the project as installation packages for different platforms.

A user needs to install IPFS, define the connection profile, and get the user certificate before using our FileWallet client. Without these dependencies, the client cannot connect the Hyperledger Fabric network and the IPFS network. Normally, the certificate is granted by the organization administrator, who takes the admission responsibility. The implemented interface of FileWallet

client is shown in Fig. 3. On the left side of the interface, the Logo and user identity is shown firstly, and three functional choices are followed: all files, downloaded, and settings. At the bottom of the left side bar, it presents the number of connected peers and the status of IPFS and Fabric networks. On the top of the interface, it provides the file search option and four functional icons: upload, new directory, import from link and refresh. The files are then shown in the main window, which performs similarly to a local file management system.

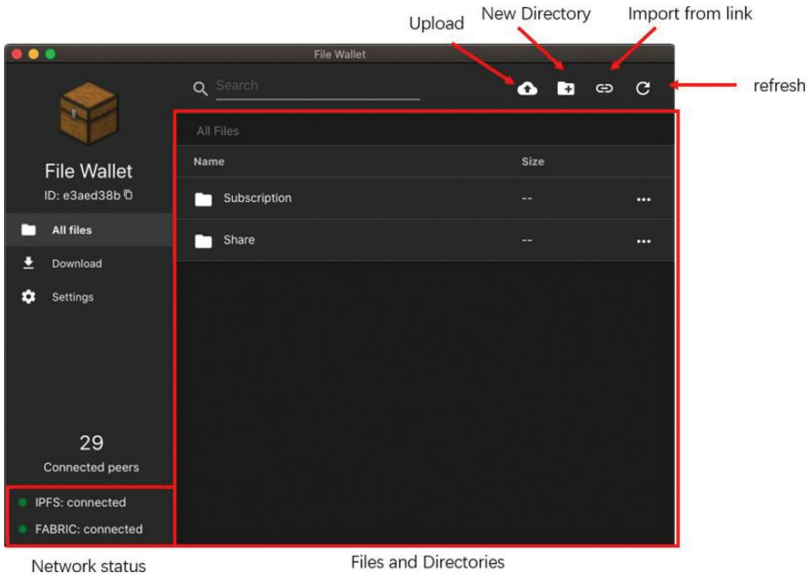

Figure 3: The index page of FileWallet client

5 Evaluation

For the purpose of evaluate the implemented system, we deploy two organizations that have one peer inside each organization and a single order node service. Each peer has a single-core CPU and 1 GB memory. The client is a 13-inch Macbook Pro that has two i5 CPUs and 16 GB memory.

5.1 Functional Evaluation

5.1.1 Directory Management

In this section, we need to focus on whether the directory name is valid and the target directory is authorized. We use the following test cases to evaluate the directory creation function. The evaluation cases are shown below:

(1) Create a directory with a valid name. The evaluation result is that the directory is created and show in the user interface.
(2) Create a directory with an invalid name, such as an empty name and a name with more than 256 characters. The evaluation result is that the user interface blocks this operation, and no proposal is sent to peers.
(3) Create a directory in an unauthorized directory. The evaluation result is that the peer refuses to sign the proposal and returns a permission error instead. The user interface displays this error to the user.

(4) Create a directory that has the same name as the existed directory. The evaluation result is that the directory fails to create because of the directory name conflict.

5.1.2 File Management

Like the directory creation function evaluation, the file addition function also needs to check the permission of the operator and whether the file name has no conflict with existed files. The evaluation cases are shown below:

(1) Add files to a directory that the user has the write privilege. The evaluation result is that the files are successfully added to the target directory.
(2) Add files to an unauthorized directory. The evaluation result is that the files cannot be added to the directory, and the user interface shows a permission error message.
(3) Add a file that has the same name as the existed file. The evaluation result is that the file cannot be added to the directory and shows a dialog to ask the user whether to overwrite the file. If the user chooses to overwrite the file, the old file will be replaced by the new file.

5.2 File Sharing

Cooperation and subscription are two ways of data sharing. Data sharing needs to ensure the operator has the permission to access the directory in order to prevent data leakage or data loss. The evaluation cases are shown below:

(1) Add a cooperator or a subscriber with an existed user id. The evaluation result is that the user has been added to the cooperator list or subscriber list of the directory and its subdirectory.
(2) Add a cooperator or a subscriber with a non-existent user id. The evaluation result is the non-existent user id error thrown by the chaincode.
(3) Add a cooperator or a subscriber with an unauthorized directory's key. The evaluation result is a privilege error thrown by the chaincode.
(4) Remove a cooperator or a subscriber from an authorized directory. The evaluation result is that the user has been removed from the cooperator or subscriber list of the directory and its subdirectory.
(5) Remove a cooperator or a subscriber from an unauthorized directory. The evaluation result is a privilege error thrown by the chaincode.

5.3 File Uploading Efficiency Evaluation

We evaluate the efficiency of FileWallet in uploading files. A certain number of files with different files are generated to be uploaded to test how our system performs.

In the first evaluation, we generate five folders, each with a different number of files, and each file has 1 MB of random content. Table 6 shows the time usage of transaction recorded and CID generation. The transaction time remains stable regardless of the file number. The transaction time is mainly affected by the network factors like latency and bandwidth. CID generation time will grow with the file number because the it requires more time to calculate the CID.

In the second evaluation, we generate ten files with size from 100 to 1000 MB. Table 7 presents the time usage of our system in uploading each single file. We can observe that the CID generation time increases when the file size raises, the transaction time is almost same about two seconds.

Table 6: The time usage with different file numbers

File number	Transaction (s)	CID generation (s)
100	2.097	13.382
200	2.141	23.057
300	2.148	38.325
400	2.737	44.958
500	2.851	55.768

The results of the above two evaluations indicate that the time of uploading files in the proposed FileWallet mainly based on the implemented network with only several peers. The time of transactions can be shortened by deploying more peers in organizations so that peers can share the endorsement workload. The client can also select the peer with the lowest latency to shorten the time consumed.

Table 7: Time usage with different file sizes

File number	Transaction (s)	CID generation (s)
100	2.059	0.896
200	2.056	1.29
300	2.203	1.918
400	2.058	3.063
500	2.06	2.959
600	2.06	3.539
700	2.058	4.217
800	2.056	4.828
900	2.059	5.796
1000	2.06	5.992

6 Conclusion and Future Work

In this study, we have proposed a decentralized file management system called FileWallet based on IPFS and Hyperledger Fabric. The system can effectively mitigate the privacy and cost issues of the existing centralized system. In FileWallet, files are no longer stored in a centralized server because in IPFS, so users can reduce upload time tremendously. Besides, the Hyperledger Fabric network enables users to share their files through channels. The Hyperledger Fabric certificates and smart contracts provide access control to our system. Finally, the efficiency of our system in uploading files is evaluated, demonstrating that the time cost of uploading files is mainly affected by devices' hardware performance rather than network conditions like bandwidth and latency.

In future work, data losing will be analyzed and addressed. If a file has few replicas in the IPFS network, it has a high risk of being lost. Hence the key to resolving data loss is creating more file replicas. How to prevent losing data in IPFS is still is a direction worth researching. Furthermore, during the development process, we discover many problems still need resolving. For example, we need to resolve the Hyperledger Fabric concurrency issue to prevent the errors when two transactions modify the same key's value. In addition to this issue, we also need to enhance

data integrity. One of the plausible solutions is developing a backup service application for users to synchronize their files to a private server automatically. As a result, other users can download the files even when the uploader's device goes offline.

Funding Statement: This work is supported in part by Key-Area Research and Development Program of Guangdong Province No. 2020B0101090005; National Natural Science Foundation of China under Grant No. 62032013, and No. U20B2046; 111 Project (B16009); and the Fundamental Research Funds for the Central Universities N182410001.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Jesse, D. D., Charles-Antoine, J. (2020). The ubiquitous digital file: A review of file management research. *Journal of the Association for Information Science and Technology, 71(1),* E1–E32. DOI https://doi.org/10.1002/asi.24222.
2. Sanjay, G., Howard, G., Shun-Tak, L. (2003). The google file system. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 29–43. New York.
3. Mirko, Z., Stefano, F., Gabriele, D. (2020). On the efficiency of decentralized file storage for personal information management systems. *Proceedings of IEEE Symposium on Computers and Communications*, pp. 1–6. Rennes, France.
4. Donie, O. (2021). Half a billion Facebook users' information posted on hacking website, cyber experts say. https://edition.cnn.com/2021/04/04/tech/facebook-user-info-leaked/index.html.
5. Juan, B. (2014). Ipfs—content addressed, versioned, P2P file system. https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf.
6. Elli, A., Artem, B., Vita, B., Christian, C., Konstantinos, C. et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15. New York, USA.
7. Yang, Z., Jiao, Y., Donald, E. P., Alex, C., Eric, K. et al. (2018). Efficient directory mutations in a full-path-indexed file system. *ACM Transaction on Storage, 14(3),* 22:1–22:27. DOI 10.1145/3241061.
8. Inglett, S. D. (1999). File system view path mechanism. US5905990, 08/880781. International Business Machines Corporation, Armonk, NY.
9. Doan, T. T., Subaji, M., Eunmi, C., SangBum, K., Pilsung, K. (2008). A taxonomy and survey on distributed file systems. *Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management*, pp. 144–149. USA.
10. BitTorrent Foundation. (2019). BitTorrent (BTT) White Paper. *Technical report*. BitTorrent Foundation. https://whitepaper.io/document/389/bittorrent-whitepaper.
11. Huang, D., Han, D. Z., Wang, J., Yin, J. L., Chen, X. C. et al. (2018). Achieving load balance for parallel data access on distributed file systems. *IEEE Transactions on Computers, 67(3),* 388–402. DOI 10.1109/TC.2017.2749229.
12. Fu, S., He, L., Huang, C., Liao, X., Li, K. (2015). Performance optimization for managing massive numbers of small files in distributed file systems. *IEEE Transactions on Parallel and Distributed Systems, 26(12),* 3433–3448. DOI 10.1109/TPDS.2014.2377720.
13. Suganya, S., Selvamuthukumaran, S. (2018). Hadoop distributed file system security a review. *Proceedings of International Conference on Current Trends towards Converging Technologies*, pp. 1–5. Coimbatore, India.
14. Satoshi, N. (2009). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.
15. Nair, P. R., Dorai, D. R. (2021). Evaluation of performance and security of proof of work and proof of stake using blockchain. *Proceedings of Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 279–283. India.

16. Zheng, Q., Yi, L., Ping, C., Dong, X. (2018). An innovative ipfs-based storage model for blockchain. *IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 704–708. Santiago, Chile.
17. Kumar, R., Tripathi, R. (2019). Implementation of distributed file storage and access framework using ipfs and blockchain. *Fifth International Conference on Image Information Processing*, pp. 246–251. India.
18. Huang, H., Lin, J., Zheng, B., Zheng, Z., Bian, J. (2020). When blockchain meets distributed file systems: An overview, challenges, and open issues. *IEEE Access, 8,* 50574–50586. DOI 10.1109/ACCESS.2020.2979881.
19. Nyaletey, E., Parizi, R. M., Zhang, Q., Choo, K. K. R. (2019). BlockIPFS–blockchain-enabled interplanetary file system for forensic and trusted data traceability. *Proceedings of IEEE International Conference on Blockchain*, pp. 18–25. Seoul, Korea.
20. Protocol Labs (2017). Filecoin White Paper. *Technical report*. Protocol Labs. *Available at:* https://filecoin.io/filecoin.pdf.