JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 40, 1-26 (2024)
DOI: 10.6688/JISE.202401_40(1).0001

BCS: Blockchain-based Ciphertext Storage Scheme Supporting Data Hierarchical Management

YUXIANG CHEN[1,2], YAO HAO[2], ZHONGQIANG YI[2],
XIAOYU GUO[2] AND CHUNXIANG XU[1,+]
[1]*School of Computer Science and Engineering*
University of Electronic Science and Technology of China
Chengdu 611731, P.R. China
[2]*Science and Technology on Communication Security Laboratory*
Chengdu 610041, P.R. China

As file management is widely used in e-government and enterprise office, the file exchange, as the main means of sharing and collaborating in office, has been far from able to meet the needs of data security. Encrypted Storage can be a solution to the limitations of existing authority control services. However, during the data sharing and exchange, the file is separated from the owner, which has the problem of insufficient or over-authorization. In this article, we propose a blockchain-based hierarchical management and control scheme for encrypted storage combined with ciphertext retrieval. It supports traceability and top-down privilege division without increasing the number of managed keys, all data are processed and authorized strictly, solving data leakage caused by over-authorization. Through performance and security analysis, we demonstrate that the scheme can better meet the data security and precise authorization requirements.

Keywords: blockchain, ciphertext storage, data sharing, fine-grained encryption, hierarchical management

1. INTRODUCTION

1.1 Background

More and more enterprise organizations choose cloud services in order to reduce computing and storage costs, document management is one of the widely used, but the security problems are becoming increasingly prominent, and security incidents are emerging one after another, resulting in economic losses and adverse effects. For example,in September 2020, United States Department of Commerce announced the removal of the application "Tik Tok" and Wechat, for their collection of citizens' data [1]. In September 2022, Northwestern Polytechnical University was attacked by the NSA (National Security Agency) for a long time and lost more than 140GB of data [2], including accounts, passwords, office-used documents and private documents, *etc*. In July 2022, service provider Didi Taxi was fined more than 8 billion yuan for leaking user data and destroying user privacy [3]. The claimed most secure cloud backup provider SOS was attacked and leaked more than 150 million users' data [4]. The security team UpGuard found that S3 storage

Received July 13, 2022; revised September 6, 2022; accepted October 20, 2022.
Communicated by Xiaohong Jiang.
+ Corresponding author.

on Amazon Cloud leaked more than 50 thousands core documents such as voter data and national strategy [5]. From statistics on data breaches in 2021, RBS (Risk Based Security) reported more than 22 billion data records leakage, and it is estimated to increase more than 5 percent in 2022 [6], the average cost will also reach a new high [7, 8].

One typical cloud storage scenario is e-enterprise, which contains several hierarchies (shown in Fig. 1). Basic employees (*e.g.* designers) can only access the data of their own class, director can access the data of their class and all their subordinate classes, and so on. Data sharing and collaboration occur frequently under such hierarchies, to make better use of data value, there often exists over-authorization in this procedure, causing data leakage, that is, users may get information of the superior classes. All in all, it is of great significance to ensure both data confidentiality and availability when users are organized in such hierarchical structures.

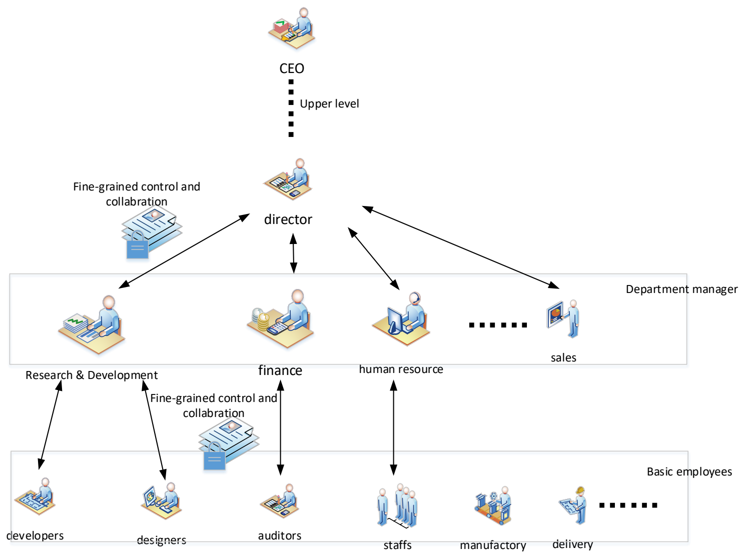

Fig. 1. Data collaboration and hierarchy.

1.2 Related Work in Ciphertext Storage

Obviously, encrypted storage will effectively alleviate data leakage problem above, a typical cloud storage structure is shown in Fig. 2, application of front-end receive request from client end and process, then store the data in database. Encryption methods can be classified into 3 categories according to the execute positions.

Position A between user and client: Position A means data will be encrypted before transmitting to client. PGP (Pretty good privacy) [9] is a typical processing method of this kind, it encrypts file data locally before uploading, but the key management is complex, users need to manually use this tool to encrypt the files before using cloud applications, which affects user experience. S. Ruoti [10] put forward secure middle layer, use UI (user interface) middle layer to cover original UI of application program, encrypt in the middle layer to replace the original function and achieve the effect of transparent

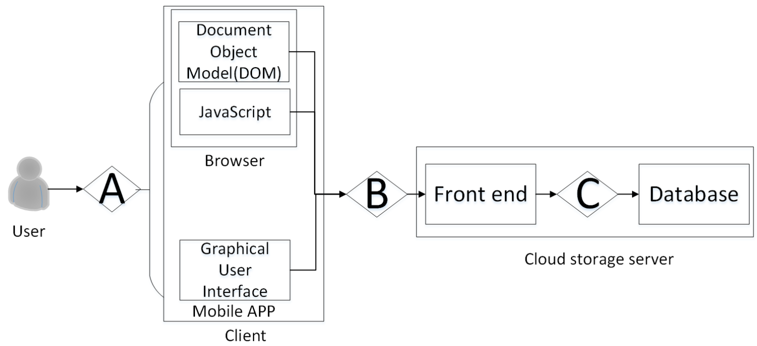

Fig. 2. Data encryption position in typical cloud server storage.

encryption. MessageGuard and Virtru [11, 12] make use of component iFrame to realize a UI middle layer, which can replace the original upload function of application to protect file, but need an extra-server to store ciphertext and lost original search function. ShadownCrypt [13] constructed a secure isolated text input/output environment based on ShadowDOM mechanism of browser. ShadownHPE [14] further put forward more stable encryption method that support text format preservation. M-Aegis [15] created a transparent middle layer to intercept and encrypt user text without affecting user's habits, but they both only support text encryption, not file encryption.

Position B between client and network: The typical representative tool for position B is CASB(Cloud Access Security Broker) [16–19], CASB encrypt and protect data uploaded to cloud, but the technology needs developers to perform reverse analysis and adaptation of the cloud service protocol, which wastes a lot of time. Once the protocol is renewed, CASB cannot identify and analyze renewed protocol, causing failure of data encryption, which is unstable.

Position C between front-end and databases: CryptDB [20] is an agent server deployed between database server and front-end, it encrypts data before uploading to database and can execute query operations directly on encrypted data, preventing malicious intenal administrators from stealing data. ARX [21] put forward database encryption system for MongoDB, supporting complicated ciphertext computing function, but both of them [20, 21] cannot prevent malicious cloud server from stealing data. Mylar [22] provides a Web framework based on Meteor JavaScript, the cloud service provider calls encryption interface of this framework to encrypt users' data. The framework assumes that service provider is trustworthy, thus cannot prevent malicious service provider. DataBlinder [23] provides distributed data access middleware for cloud service provider, still untrustworthy to users. The above protection methods make encrypted data lose availability, which is difficult to develop and utilize, further, from the perspective of information security, the current management and control of data still faces a series of challenges.

In a document, there are different authorization levels for different users. When the file is shared, the content of the file will exist out of the control of the data owner. The data of different authorization levels in the same file has hierarchical control requirements in the process of data sharing, exchange, storage, *etc*. [24, 25]. If there is insufficient protection of the data authorization scope, it will lead to the risk of data leakage. The current encrypted storage is still based on the directory level and file level, it lacks metadata to record the data blocks in the file. The same file can only be encrypted as a whole, but different parts of the file cannot be distinguished and targeted to protect. This unified coarse-grained encryption method encrypts all the data of the same file that has different authorization categories, which wastes CPU computing resources and it also does not meet the requirements of data hierarchical control. If it is necessary to divide ciphertext permissions in different areas of a file, the existing method can only encrypt the file multiple times with different keys, thus generating multiple files, which is not conductive to users and programmers to manage these files and wastes a lot of storage resources.

1.3 Blockchain-Based Security and Authorized Data Sharing

Blockchain is the underlying technology of crypto-currencies, first introduced by Satoshi Nakamato [26]. It simplifies the transfer process of value and trust, makes data records have the advantages of traceability and anti-tampering, which attracted a lot of attentions, especially in the field of information security in different scenarios such as artificial intelligence, edge computing, industrial internet, *etc*. [27,28,30], which demonstrate the advantages and availability of applying blockchain to data security sharing and exchange. Blockchain is a complicated system, which can be regarded as a fundamental infrastructure, Many schemes use it as a trusted database to read and write data, whereas it consumes much more computational overhead when it comes to writing including the hash, signature, consensus algorithm, *etc*. Furthermore, multi-node back up means each ledger stores multiple copies, and thereby also wastes storage resources compared with normal storage. To sum up, even writing a small amount of data, the overhead increases exponentially, resulting in a decline in the performance of data service, such as an increase in response delay. Table 1 compares the differences of some typical works in the field of (blockchain-enabled) data secure sharing.

Table 1. Related work on blockchain-based secure and authorized data sharing.

Reference	Related work	Authorized data sharing	Blockchain-based security
He *et al*., 2020 [31]	Access Control for sharing the data	Attribute-based Hierarchical scheme	None
Gao *et al*., 2020 [32]	Trustworthy secure data sharing	Fine grained based scheme	None
Sun *et al*., 2020 [33]	Data encryption scheme for access	IPFS-based encryption scheme	Generic blockch*et al*. evidence for behaviors
Qin *et al*., 2021 [34]	Access control strategy for data sharing	Multi-authority by using Shamir secret sharing	Consortium blockchain supporting multi-authority
Zuo *et al*., 2021 [35]	Sharing data securely in cloud without any trustworthy parties	Ciphertext policy attribute based encryption scheme	Generic blockchain for traceability
Athanere *et al*., 2022 [36]	Hybrid approach for multi-authority data sharing	Multi-authority attribute-based encryption approach	Blockchain-based cross domain collaboration
Zhang *et al*., 2022 [37]	Encrypt speech data based on blockchain in a distributed way	Ciphertext policy hierarchical attribute-based encryption to support data sharing	Generic blockchain for traceability and controllability in a distributed way

1.4 Problem Statement and Our Contributions

1.4.1 Problem statement

We aim at typical scenes such as *e*-office, *e*-government, and so on, which contain several hierarchies as shown in Fig. 1. The security issues for encryption methods are subdivided as follows,

- The simple overall encryption method in file management has the problem of too coarse granularity, which leads to over-authorization in the data flow and increases the risk of data leakage.
- In the current application scenarios such as e-government, corporate office, *etc*. It is difficult to divide ownership and security responsibilities in data circulation, which makes participating entities "unwilling" and "dare" to share data, hindering the value of data.
- Existing methods to solve data encryption and authority management have the problem of wasting computing and storage resources, which is not conductive to users and programmers to manage data.

1.4.2 Our contributions

We proposed blockchain-based ciphertext storage scheme supporting data hierarchical management, its novelty is under the background of e-government, e-enterprises, *etc*., where data sharing and collaboration is frequent between different departments, superiors and subordinates, but users at different levels in different departments have different read permissions in the data collaboration, there is often the problem of excessive authorization, thus causing data leakage. How to balance accurate fine-grained authorization and prevent excess information leakage is one of the difficulties, our contributions are threefold below:

- Firstly, based on random keys and data tags, we implemented key-based hierarchical management of different data blocks of files. Random keys are derived from root keys, random numbers and file attributes, data blocks with different permissions are used different keys to encrypt.
- During data transferring, authorized high-security users can derive keys of low-security data blocks to decrypt the data, but they cannot derive the keys of higher-privileged data blocks, thus realizing top-down authority division. Further, users don't need to store multiple keys, preventing uncontrollable data leakage and reducing key management costs. Metadata like data tags are organized in a tree structure to quickly extract and process ciphertext blocks.
- In order to enhance the trust of the system, we introduce blockchain to store metadata of ciphertext, data confirmation and circulation information. Under the premise of taking into account the efficiency, prevent the loss of ciphertext data caused by single point of failure, and divide the data responsibility to assist the confirmation of rights.

The rest of the paper is organized as follows: we first present an introduction to blockchain-based ciphertext storage system. Then a fine-grained access control encryption based on key derivation is presented. We further present the effect of hierarchical control of data transfer, efficiency of encryption and blockchain operation, and provide security analysis of our schema. Finally, the conclusion will be drawn.

2. APPLICATION BACKGROUND AND MODEL

2.1 Structure of Ciphertext Search System

The structure of blockchain-based ciphertext search system is shown in Fig. 3. The client of ciphertext file system uploads and downloads files to the ciphertext file storage system, and performs encryption and decryption operations in the background of the client during uploading and downloading. When the client uploads the ciphertext, it also establishes a cipher index of keywords list in the cipher file system processing layer for searching the ciphertext.

When the client initiates the sharing operation, it distributes the file key to the shared user's client through the secure channel and notifies the shared user's client. After the shared user's client receives the sharing, its client updates the cipher index to the cipher file processing layer. The cipher file processing layer establishes a cipher index for the client, provides storage, update and query of the index list.

When user register, the key management sub-system provides initial keys such as master key, search key, *etc*. Meanwhile, it provides the file key to the shared user when the client(user) executes the sharing operation.

Blockchain exists as a trust enhancement facility in the structure, user client writes the metadata of encrypted files into the blockchain in case he loses encrypted files stored on big data platform, further, he can publish the data fingerprints to the blockchain to confirm data asset right. Meanwhile, the authority(key management) write the data transfer record into the blockchain to support follow-up traceability and division of responsibilities.

To better make use of advantages of the blockchain and take into account the practicality, we actually divide data storage into on-blockchain and off-blockchain storage, of which metadata belongs to the on-blockchain storage for its relatively small size, large-size file data remain in off-blockchain storage.

2.2 Fine-Grained Encryption Model of Files

In the design of data hierarchical management and control model, we consider personnel authority, data authorization and data owner's control authority over published data at the same time. The encryption model is shown in Fig. 4. When data owner uploads a file, the client's background divides the file data into blocks according to the data labels, judge the data tags and deriving corresponding data block key. Finally, encrypt blocks of data with different classified labels separately before uploading.

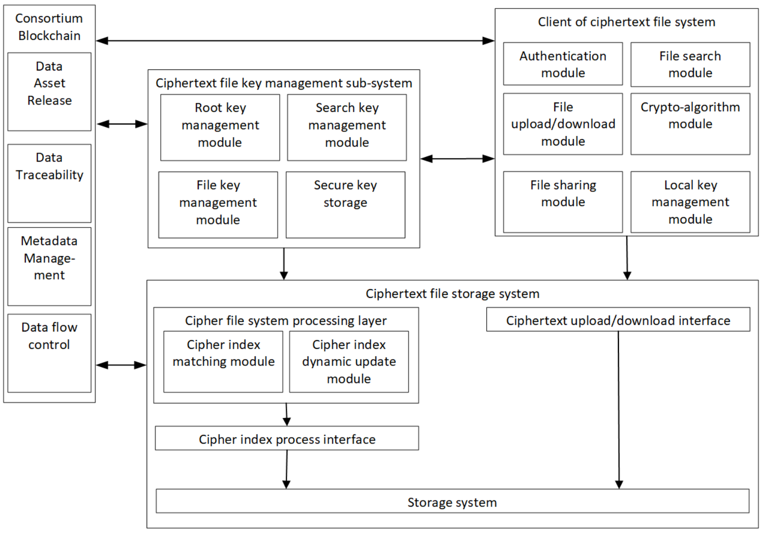

Fig. 3. Structure of blockchain-based ciphertext search system.

2.3 Fine-Grained Authorization and Decryption Model of Files

When the encrypted data block is shared to users with different permissions. Authorized users can only decrypt data blocks of the same and lower security level. As shown in Fig. 5, take the management with 4 class data block permissions as an example. The 1st authorized user has the readable permission of all 4 types of data blocks. 2nd authorized user has the readable permission of level 2 and below security level data blocks, totally 3 kinds and so on.

3. FILE HIERARCHICAL MANAGEMENT AND CONTROL SCHEME BASED ON RANDOM KEY

3.1 Hierarchical Key Management

In terms of key management, personnel authority, data security level, authorization authority and data owner's control authority over published data are considered at the same time. The key derivation and management relation is shown in Fig. 6, Each user has a self-registered random root key RK generated by the function Rand, which can be self-generated or issued by an authority, the RK can be used to derive the file key FK ($FK = Hash(RK \| filename)$) of the published file. At the same time, the user obtains corresponding level of authorized key LK_n (n = 1, 2, ..., n) from an authoritative center (key management sub-system), the key is transmitted through a secure channel such as digital envelope, HTTPS, *etc*. The user only needs to store the personal root key RK and his highest-level authorized key, and does not need to store other keys. The secondary authorization key, file key, and authorization data block key can all be generated through derivative calculations. The derivation method of the nth level authorized key is $LK_n = Hash(LK_{n-1} \| n-1)$, and the corresponding derivation method of the nth level

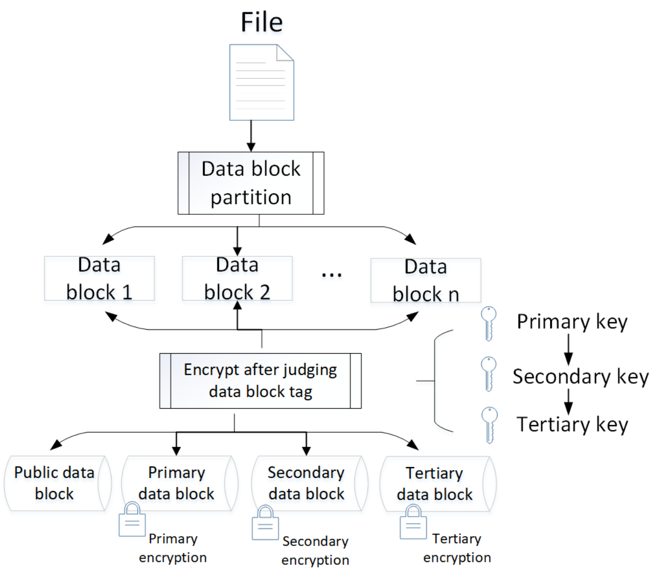

Fig. 4. Fine-grained encryption model.

data block key is $FK_n = Hash(LK_{n-1}||LK_n||DataLale)$, where LK_n is the corresponding authorization key, FK_{n-1} is the upper-level data block key. DataLable is the security level identification of the nth level data block, and Hash is the hash function. It can be seen that when the user has the root key and the authorized key LK_n, only the data block key of the same level or lower security level can be calculated to process corresponding data, and the key of higher security level data cannot be calculated. When a user authorizes the specified data content of a file to other users, only the data block key of the corresponding level needs to be given through the secure channel to realize the data authority division.

3.2 Hierarchical Encryption of File Content

In terms of file encryption, the file security level is equal to the highest security level of its content. We still take 4 level data block authority management as an example(shown in Fig. 4). A 1st level authorized user needs to upload files with different levels of information. Assuming that he has three different levels of files (FileA, FileB, FileC) needs to upload, which correspond to the file keys FKA (1st level), FKB (2nd level), and FKC (3rd level). According to the operating rules, 1st level files can contain 1st, 2nd, 3rd and public data blocks. 2nd level files can contain 2nd, 3rd and public data block. 3rd level files can only contain 3rd and public data block. The user first calculates the file key of each file according to the key derivation step to get file keys (FKA, FKB, FKC). For a 1st level file FileA, the hierarchical encryption key of each data block inside it is generated by the file key FKA ($FK_A = Hash(RK||Filename)$). There can be up to 4 levels of data blocks in FileA, 1st, 2nd, 3rd and public information. The user then further calculates the hierarchical keys of the three data block levels:

$$FK_{A1} = Hash(FK_A||LK_1||Datalable1), \tag{1}$$

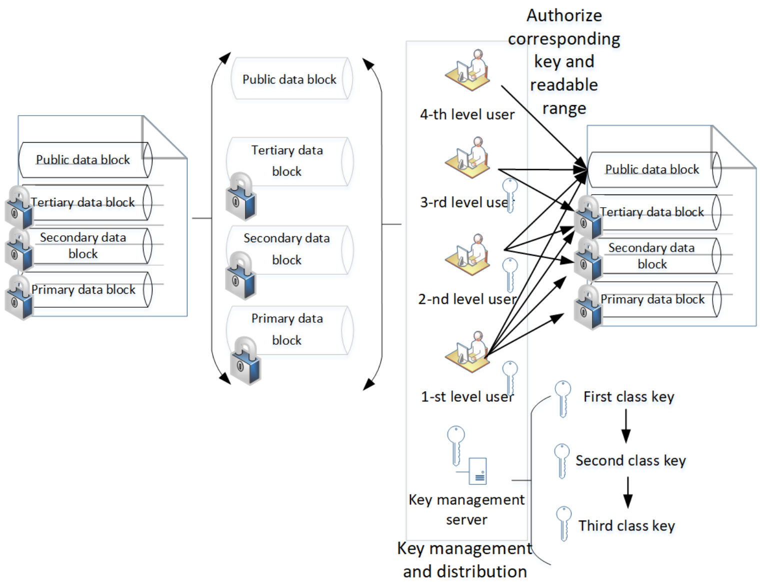

Fig. 5. Fine-grained decryption model.

$$FK_{A2} = Hash(FK_{A1}\|LK_2\|Datalable2), \tag{2}$$

$$FK_{A3} = Hash(FK_{A2}\|LK_3\|Datalable3). \tag{3}$$

In the above calculation, the user holds the level key LK_1 issued by the authority, and derive the subsequent level key by using the formula $LK_n = Hash(LK_{n-1}\|n-1)$ shown in Fig. 6.

Finally, the user encrypts the contents of FileA,

$$\begin{aligned} A_C &= (C_{A1}, C_{A2}, C_{A3}) \\ &= (Enc(File_{A1}, FK_{A1}), Enc(File_{A2}, FK_{A2}) \\ &\ Enc(File_{A3}, FK_{A3})). \end{aligned} \tag{4}$$

For the 2nd file FileB, there can be at most 3 types of data blocks with different security levels, namely, secondary, tertiary and public, the user calculate hierarchical keys of two data block levels respectively:

$$FK_{B2} = Hash(FK_B\|LK_2\|Datalable2), \tag{5}$$

$$FK_{B3} = Hash(FK_{B2}\|LK_3\|Datalable3). \tag{6}$$

Then encrypt the contents of FileB:

$$\begin{aligned} B_C &= (C_{B2}, C_{B3}) \\ &= (Enc(File_{B2}, FK_{B2}), Enc(File_{B3}, FK_{B3})) \end{aligned} \tag{7}$$

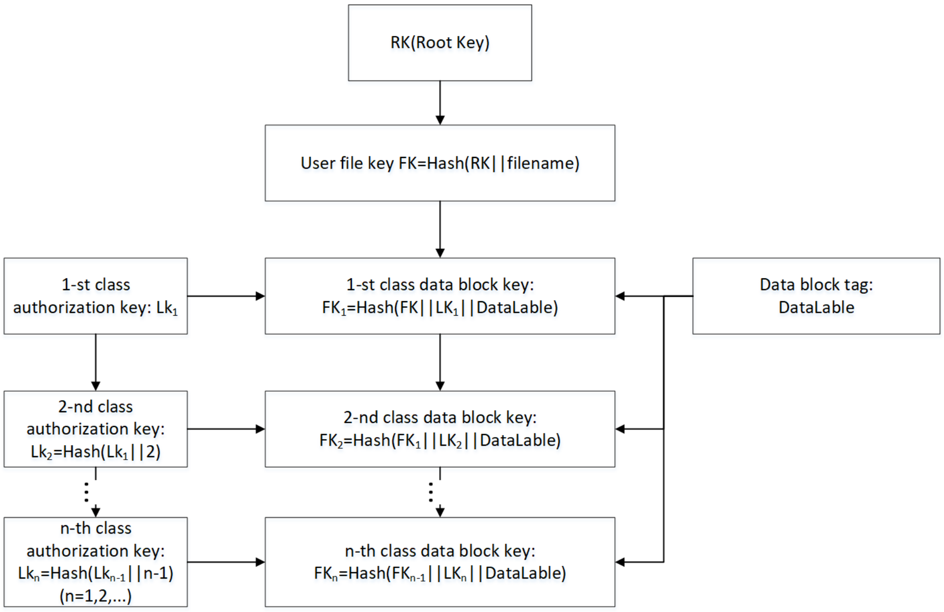

Fig. 6. Key derivation relation and management.

Similarly, for 3rd level file FileC, only 2 types of blocks exist, *i.e.*, 3rd level and public data blocks, so calculate the corresponding 3rd level key,

$$FK_{C3} = Hash(FK_C \| LK_3 \| Datalable3) \tag{8}$$

and then encrypt the contents of FileC:$C_C = Enc(File_{C3}, FK_{C3})$ In the above calculation, the 4th type data (public data block) is not processed and still kept in plaintext.

When sharing files to different authorized users, the sharer only needs to give the corresponding data block key FK_i (i = 1, 2, ..., n) with the highest authority through the secure channel.

3.3 Metadata Organization and Management Structure

When the file is encrypted in blocks by the creator, because the size of each file data block is variable, in order to facilitate recovery, the metadata information of the entire file is organized hierarchically in a tree structure and stored independently (such as metadata management server, independent area in file header, *etc.*) for query, when the user decrypts the file blocks, he will find the corresponding permission data block and derive the correct key to decrypt. The metadata mainly includes data lable, authorization range, offset address, encryption algorithm, *etc.* as shown in Fig. 7.

It is worth noting that metadata occupies far more less space than ciphertext itself. We encrypt it with the corresponding file key and store it on the blockchain, thus taking advantage of its multi-node backup feature to prevent ciphertext data from being unrecoverable due to the loss of metadata.

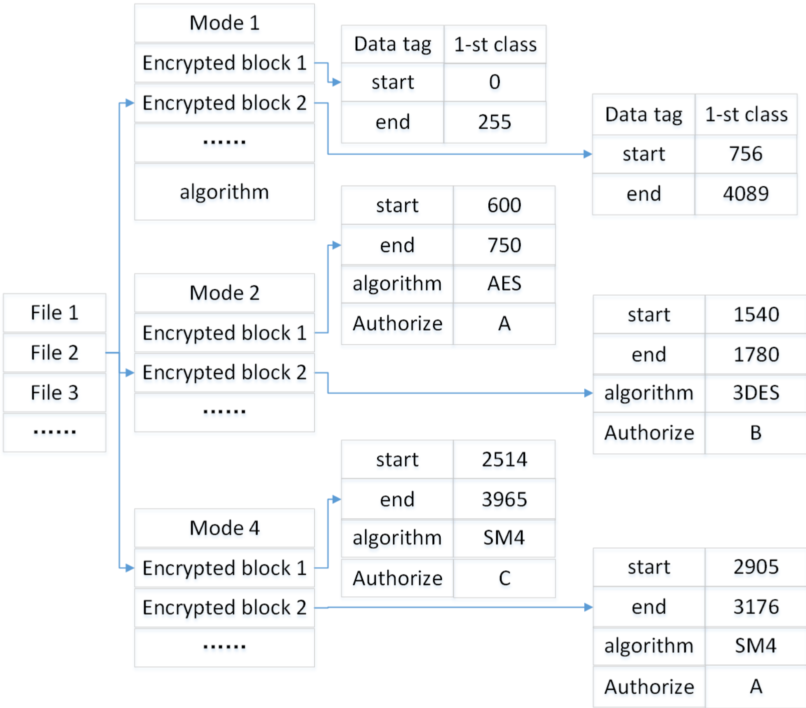

Fig. 7. Metadata management structure.

3.4 Hierarchical Decryption of File Content

When the shared user needs to decrypt, he can retrieve file's metadata from blockchain and verify its validity, then he can recover all the encryption keys (including FK_{i+1},FK_{i+2},...,FK_n) of the lower-level data blocks in the file according to the key LK_i of his authorized level and shared data block key FK_i. The specific process is as follows:

Step 1: Meta-data parsing process The user queries the blockchain for meta-data information based on the unique identifier of the file. Firstly, he/she verifies the creator's signature of the metadata returned by the blockchain, after that, he parse the meta-data to obtain the data block information at all levels of the file, including the identification of each ciphertext block, the algorithm used(encryption mode), security level, starting/ending positions, *etc*. Extract the public parameters required for decryption for subsequent steps to call.

Step 2: Key derivation process The user with class i already has his level key LK_i when register. After receiving and parsing the file, he/she gets the public parameters of the file data block and shared file class key FK_i through secure channel(such as digital envelope) , and uses these parameters to calculate the lower-level key $LK_n = Hash(LK_{n-1} \| n-1), n = i+1, i+2, \ldots$ locally step by step.

After getting all the authorized level keys $LK_n, n = i, i+1, \ldots, l$ (l is the number of classifications in the system, and satisfy $i < l$). He/She further calculate the data block keys $FK_n = Hash(FK_{n-1} \| LK_n \| DataLable), n = i+1, i+2, \ldots, l$ (already has FK_i). After getting all the data block key $FK_n, n = i, i+1, i+2, ..., l$, the key derivation process is over.

Step 3: Local decryption in the client After the above preparation, the user can decrypt the data blocks one by one with his key collections, suppose the received File consists of l encrypted data blocks with level identification ($file = C_1, C_2, ..., C_l$, $0 < i \leq l$, C_i means the encrypted data block i while P_i corresponds to the plaintext of the data block i). He/she can decrypt the encrypted data blocks,

$$\begin{aligned} file &= (C_1, C_2, ..., C_{i-1}, Dec(C_i, FK_i), Dec(C_{i+1}, FK_{i+1}), ..., Dec(C_l, FK_l)) \\ &= (C_1, C_2, ..., C_{i-1}, P_i, P_{i+1}, ..., P_l). \end{aligned} \tag{9}$$

From Eq. (9), we can see that the user can only decrypt the data blocks $(C_i, C_{i+1}, ..., C_l)$ he has been authorized, while the rest blocks of the file $(C_1, C_2, ..., C_{i-1})$ remain encrypted, because he can't reversely derive the superior class keys $(LK_{i-1}, LK_{i-2}, ..., LK_1)$ of higher security level through key derivation mechanism shown in Fig. 6.

Finally, the shared user can use the metadata management structure to extract all the authorized ciphertext blocks and use the recovered data block key FK_i to decrypt the file data blocks ($File = Dec(C_i, FK_i)$) within his authorization. A case of authorization and reorganization of data blocks in a file is shown in Fig. 5, that is, the user's readable permissions are not higher than the authorized permissions of the file data blocks.

4. PERFORMANCE AND ANALYSIS

4.1 Performance

We verified our scheme from the perspective of engineering, based on the structure of ciphertext search system shown in Fig. 3, we further integrate our hierarchical management scheme into the prototype.

We take markdown, a lightweight markup language and one of the most commonly used document types, as an example and parse it. Fig. 8 shows that we login into a client and uploads a markdown file test_yzq.md. We can see that the plaintext is divided into 4 parts, that is, 4 types of data blocks, the security level from high to low are level 1, 2, 3, 4, of which level 4 belong to public block. if the user needs to upload, each block except the head metadata will be encrypted with a different key, the keys are organized as shown in Fig. 6. Finally, the uploaded file in the cloud (storage system) will be completely garbled except metadata. So we don't show the demonstration effect of full ciphertext data (garbled). We select 2 level 2 shard user ID in the client interface, at this time, the trusted key management center and cloud storage will authorize the shared user. We log into the shared user's account on another terminal, accept and download the markdown file. The decryption effect is shown in Fig. 9. It can be seen that the 2nd shared user can only decrypt the 2nd level data block and below.

Similarly, the 3rd level shared user can only decrypt the 3rd level block and below (shown in Fig. 10). The rest higher level blocks are still presented in ciphertext garbled.

To sum up, we have realized the hierarchical control of ciphertext data flow.

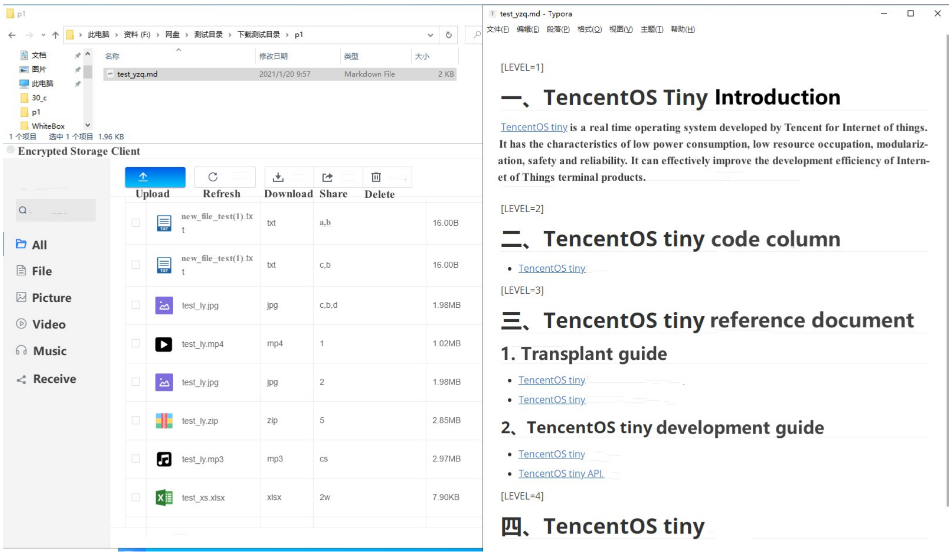

Fig. 8. Original plaintext file before uploading.

4.2 Efficiency Analysis

The efficiency of the scheme is mainly affected by the following aspects: efficiency of encryption algorithm, writing and query efficiency of blockchain.

4.2.1 Encryption efficiency

The technical route we choose is symmetric encryption algorithm, we noticed that a fair amount of encrypted storage work is based on ABE (Attributed based encryption),for example, shown in Table 1. ABE is essentially an extension of public key encryption, for example, schemes base on RSA [35, 36], ECC (Elliptic Curve Cryptography) algorithm [34, 37, 38], *etc*. Since the security assumptions of symmetric encryption, RSA and ECC are different, the key lengths are also different under the same security level. According to the judgment of NIST (National Institute of Standards and Technology) [39], there is an approximate analogy between the key strength of asymmetric encryption and symmetric encryption, shown in Table 2, that is, the difficulty of brute force cracking of data encrypted with different key lengths of each line is equivalent. For example, a 1024-bit RSA key has the same strength as an 80-bit symmetric encryption key or a 160-bit ECC key. At present, the commonly used RSA secret key security strength is 2048 bits, and NIST suggests that after 2030, at least 3072-bits of RSA secret keys (corresponding to 128-bit symmetric keys or 256-bit ECC keys) should be used to ensure the security.

From the comparison shown in Table 2, our route is superior to other schemes in terms of security and key length. In fact, whether it is ECC or RSA, it is mostly used for handshake-exchange session keys and authentication, such as TLS (Transport layer security)/*ssl*(Secure sockets layer) authentication or encrypting symmetric key data by means of digital envelope method, which is a typical *kb*-level data processing. Once MB or even GB-level file data is involved, efficient symmetric encryption is usually adopted.

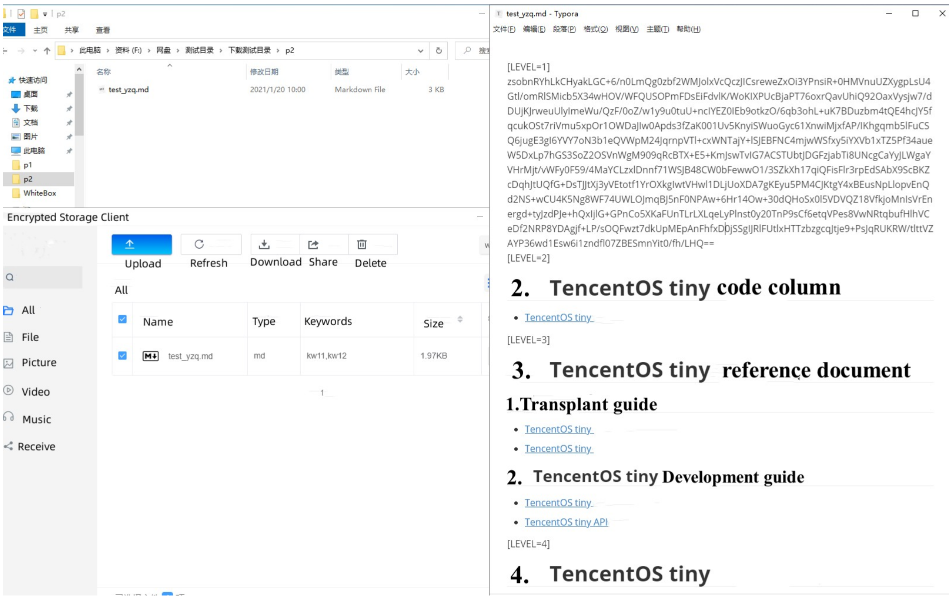

Fig. 9. 1st level authorization decryption after data sharing.

We have not seen the use of asymmetric encryption to process large file data (MB-level or above) in practical scenarios yet.

Table 2. Key length corresponds to security level.

Security level (bits) of symmetric key length (our scheme)	Key length (bits) of RSA mechanism [35, 36]	Key length (bits) of ECC mechanism [34, 37, 38]	Key length comparison: our scheme/RSA/ECC
56	512	112	1:10:2
80	1024	160	1:14:2
112	2048	224	1:20:2
128	3072	256	1:24:2
192	7680	384	1:40:2
256	15360	512	1:60:2

We adopted the most commonly used commercial cryptographic algorithm SM4 (block cipher algorithm) and SM3(Hash algorithm) to demonstrate the encryption efficiency of our hierarchical scheme. Considering that encrypt/decrypt operation is executed on user client, the computer we use is configured as Intel-i7 with 16GB memory. The consuming of encryption is shown in Table 3, and the average time consuming is 305.67 Mbps. Consider that an English letter occupies 1 byte of space (while a Chinese character occupies 2 bytes), files with more than 50 million English letters can be encrypted in one second. Therefore, client-side encryption won't affect user experience.

Further, consider the most extreme case in Fig. 6, if a 1st class authorized user needs to decrypt the nth class data block, it takes $2n$ times hash operations and one decryption

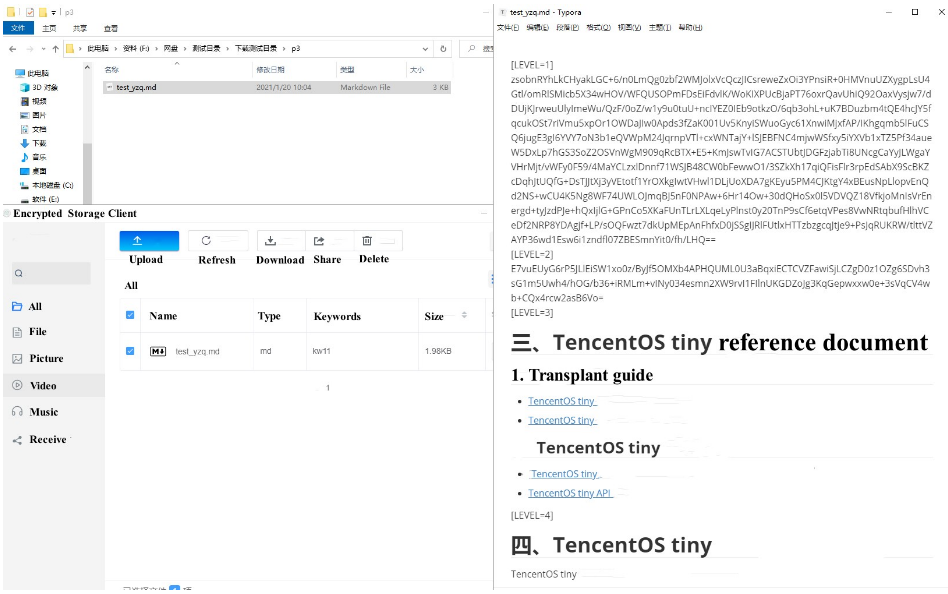

Fig. 10. 2nd level authorization decryption after data sharing.

Table 3. Encryption efficiency of 100M data.

Times	Time consuming (ms)	Times	Time consuming (ms)
1st	396	6th	314
2nd	330	7th	336
3rd	341	8th	308
4th	308	9th	310
5th	343	10th	324

operation from key derivation to decryption. The length of sm4 key is only 128 bits, hash calculation overhead can be approximately ignored.

4.2.2 Blockchain efficiency

Considering that in the data security sharing and exchange, it is necessary to ensure the credibility of the authority, as well as the distributed consensus and ledger's authorization of reading and writing, we adopt the typical alliance blockchain [27,28], hyperledger fabric. The platform not only meets the authorization of writing and reading, but also has strong scalability and high efficiency in the consensus algorithm when compared with public chain and private chain. We constructed the blockchain experimental environment to verify the feasibility of the scheme, the test environment of the blockchain is shown in Table 4, which includes 3 servers with 16GB memory, containerized services are deployed on each server, each has an endorser and an orderer node.

The reproduction of this part is easy to implement. "Hyperledger fabric" [29] can be easily downloaded from the Internet together with coursebook, readers just need to deploy it according to the instructions, find its read/write interface, and can further simulate the reading, writing and querying of metadata (usually *kB*-level).

Table 4. Test environment of blockchain.

Server	CPU	Memory(GB)	Containerized service
Server 1	4	16	1 endorser and 1 orderer node
Server 2	4	16	1 endorser and 1 orderer node
Server 3	4	16	1 endorser and 1 orderer node

In the above experimental environment, we call the read/write interface to realize the functions of publishing(sharing) and querying data on the blockchain. We use Jmeter test tool to encapsulate the data release and query interfaces with HTTP, and simulate user writing and query behavior with 50 and 100 concurrent calls respectively, the test results are shown in Table 5, which is in line with expectations.

Table 5. Latency of client user/authority requests.

Operation	Number of concurrent users	TPS (Transactions per second)	average latency (ms)
Data release	50 100	94.5 174	529 574
Data retrieval	50 100	55.8 108.3	896 923

Fig. 11 compares our BCS scheme with other blockchain-based data sharing system [40, 41]. With the number of concurrent user requests increases, so does the average latency, and the magnitude of the delay is basically the same. However, the storage resources of blockchain are limited by computational overhead. Unlike other schemes [40, 41], which store plaintext or ciphertext of equivalent size, our scheme costs less, it only stores metadata information and share credentials of ciphertext, which is superior to schemes [40, 41] in security, further, we can see our scheme has lower delay when concurrent requests are large (100 users), so it is more effective in handling large amounts of data.

4.3 Security Analysis

We think that the fine-grained hierarchical data should meet 3 basic security needs: confidentiality, integrity and availability. Confidentiality guarantees that only authorized users can read. Integrity indicates that the data is not tampered with in the exchange. Availability denotes that data services are available when users need them. In relation to typical security models, we assume that an adversary can't break an encryption or signature, then discuss the threat models, including threats to confidentiality, availability and analyze whether these models threats to our design.

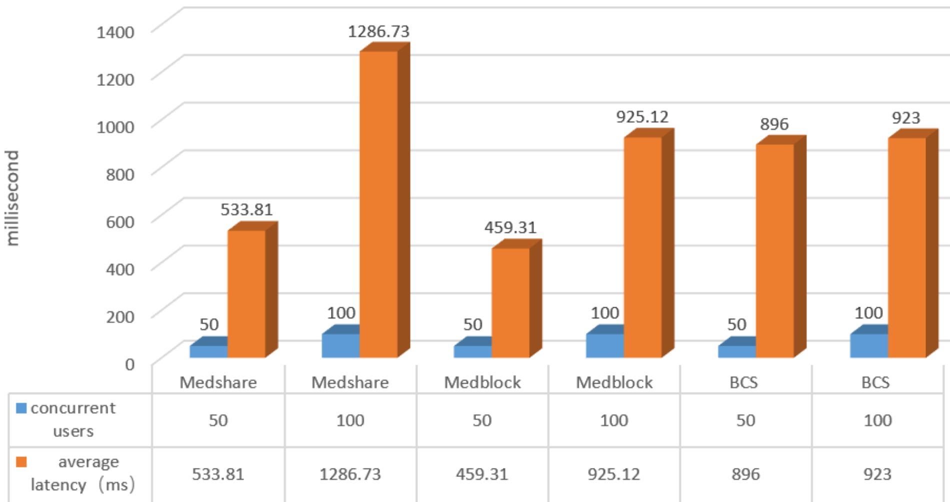

Fig. 11. Latency comparison of different scheme.

4.3.1 Hash function and security model analysis

As we designed in Fig. 6, we use hash function to realize the top-down authorization logic in the process of key derivation to fit the application scenario. Hash function is a one-way function, given an input x, it will calculate the corresponding output $H(x)$. The main characteristics of hash function are:

- The input x can be a string of any length.
- The length of the output result $H(x)$ is fixed.
- The process of calculating $H(x)$ is efficient, that is, the time complexity of calculating $H(x)$ is $O(n)$ for string x with length n.

Further, hash function has the following security properties:

Unidirectional: For any given hash value h, it is computationally infeasible to find x that satisfies $H(x) = h$.

Weak collision resistance: For any given message x, find a message y ($y \neq x$ and satisfying $H(x) = H(y)$) is not feasible in calculation.

Strong collision resistance: It is computationally infeasible to find any pair message $(x, y), x \neq y$ satisfying $H(x) = H(y)$.

Theorem: If an internal adversary can destroy the confidentiality and integrity of the upper-class data in a file by obtaining the upper-level key, then the hash function's resistance property is broken.

***Proof*:** Assume that an adversary A can forge a hash value without knowing the input key details; Then, we construct another algorithm to break hash function's collision resistance property.

We regard hash function H as an oracle with randomness. Each encrypted data block $Enc(FileBlock, FK)$ (with level key LK) can only query H once.

Initialization: Input a secure parameter λ, the challenger C generates public parameter "para"(includes class, data label, etc), send them to the adversary A.

Query: Adversary A adaptively issues the query to challenger C as follows: A can ask for information about any data, such as $j-1$th encrypted data blocks $Enc(File\ Block_{j-1}, FK_{j-1})$ and the hash value of the superior class LK_j ($LK_j = Hash(LK_{j-1} \| j-1)$). C generates $hash_j$, sends it to A.

End: Receiving a set of key information other than Lk_{i-1}, the adversary A outputs a $hash_i$ under the ith class information. A wins if $hash_i$ sent by A is valid and differs from that of A while the challenger C doesn't reject it.

The outcome denotes that the adversary can destroy the confidentiality and integrity of upper class key Lk_{i-1}, and can further decrypt data blocks of higher class, this obviously contradicts the anti-collision property of hash function.

4.3.2 Security of hierarchical data control and key management

In order to make the encrypted data hierarchical management and control more in line with application scenarios, for example, in companies, department members collaborate to process and store data in the cloud, the leader needs to be able to decrypt and view all the files of his subordinates under his jurisdiction on the cloud disk, but not vice versa. As mentioned in the background of the paper, considering that data is not stolen by the management party, all data is stored in the cloud managed by third parties in an encrypted way. Considering the problem of data leakage caused by over-authorization, it leads to the fine grained permission division and cooperation requirements based on encryption. Different types of data need to take into account availability while being classified and protected. In order to realize fine-grained authority division, key management is divided into two lines: user level and data level, corresponding to the left column and middle column in Fig. 6 and random factors are introduced(That is, data labels with path). The data block key can only be derived from top to bottom based on the input of superior data block, user authorization key and random factor, but can not be reversed. In this way, encryption of each block with different keys can be realized without storing a large number of keys, which increases the difficulty of adversary cracking, prevents data leakage from getting out of control when a single key is cracked. When sharing, only the authorized block key needs to be given through secure channel and the shared user can calculate the low-level authorized key by himself, but can not deduce keys beyond the authorization, thus laying a foundation for accurate authorization.

In terms of one-way privilege division from top to bottom, privilege division based on key means that superior users can get subordinate keys, while subordinate users cannot get superior keys. Traditional methods will make users with higher authority get all the block keys, which will cause huge management and storage overhead in big data scenarios. In view of the above properties of hash function, key derivation management will be unidirectional from top to bottom. That is, the superior key can deduce the subordinate key, but the subordinate key cannot deduce the superior key, which is valuable in use.

4.3.3 Trust enhancement and traceability

Besides, we realize data confirmation, behavior traceability, multi-node backup, integrity, accuracy and timeliness by virtue of blockchain [26, 27]. The key to the anti-tampering of blockchain lies in the chain structure of its data layer, shown in Fig. 12, which supports conformation, traceability, *etc*. The basic unit of the chain structure is block, which contains block header and block itself. The block stores transactions like metadata, "ransfer voucher" submitted by user, authority, etc, while block header contains hash of previous block, which can be understand as a pointer to the parent block, besides, it contains merkleroot, production time of current block (time stamp), *etc*. Merkleroot summarized the transactions in current block through hash function step by step, which can be used to quickly verify the contents in the future. The security of chain structure can also be reduced to the unidirectional, weak collision resistance and strong collision resistance of hash function shown in Section 4.3.2.

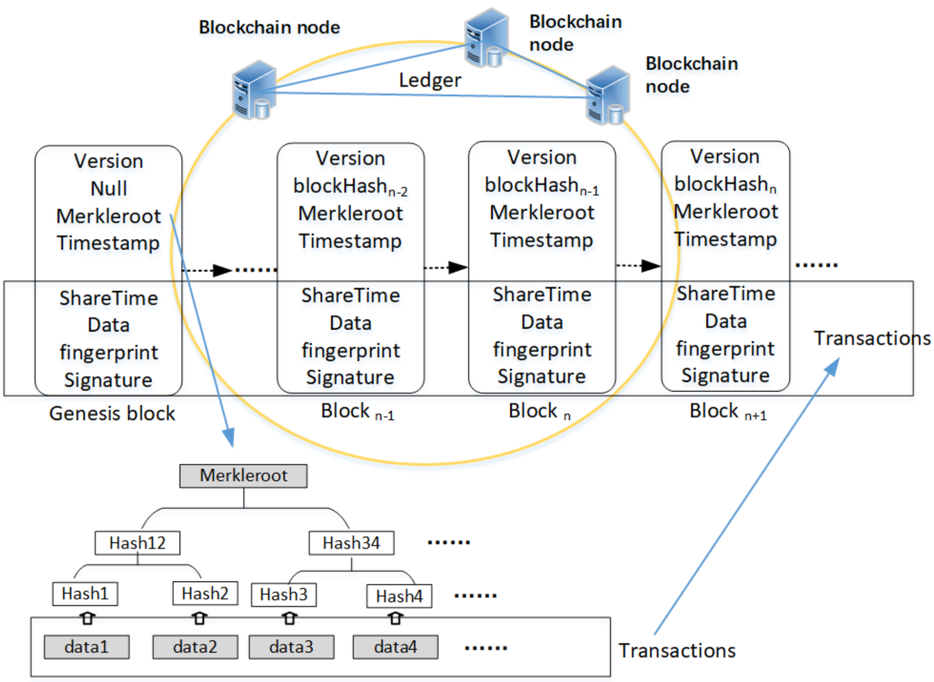

Fig. 12. Latency comparison of different scheme.

The security analysis of hierarchical data circulation is specifically as follows:

1. Metadata of files is written into the blockchain by each user as a backup to efficiently restore the ciphertext. Moreover, considering blockchain node is relevant public, user encrypt each metadata with corresponding file key FK, which can only be retrieved by userID and file unique identifier.

2. Every time a user initiates a sharing operation, the "transfer voucher" (including key distribution, time, *etc*.) is endorsed(signed) by the key management (authority) and written into the blockchain, so that the traceability of data flow behavior has credibility.

3. Blockchain has the characteristic of multi-node backup, it will decentralize management pressure such as query, insert when compared with TTP (Trusted Third Party), each node can provide basic metadata management service. The scheme can also partly get rid of single point failure faced by TTP in terms of metadata management, and divide the data responsibility to assist the confirmation of rights.

To sum up, we realize the accurate authorization, responsibility division and management of data, guarantee the security.

4.4 Application and Advantages

The method designed above only encrypts plaintext itself, which can be summarized as,

$$c \leftarrow SKE.Enc(K,d). \tag{10}$$

K stands for key set, SKE is the abbreviation of key encryption, and Enc means encryption, c represent ciphertext result. However, encrypted data storage faces problem of availability, if download the encrypted data to the client and then decrypt and search, the search efficiency will be low. If decrypt the ciphertext and search plaintext on the server will reduce security. It is usually combined with searchable encryption technology to balance security and availability [42–45]. That is to say, several keywords of a document are set or extracted, and the keywords are processed by a searchable encryption algorithm to generate the ciphertext index, which is a mapping from keywords to document ID. Only the encrypted search token generated by keywords searches the cipher index and successfully obtains the file ID, users can download the ciphertext file and then decrypt the corresponding data block according to their own authority.

The distributed security storage system for big data platform, which is characterized by fine-grained control, ciphertext storage and processing, is to be explored and demonstrated in the fields with big data platform such as e-office and medical data sharing. Take e-office data sharing as an example, the cloud big data storage platform provides fine-grained encryption storage protection and controlled sharing mechanism for user data in scenarios such as e-office, as shown in Fig. 13.

Table 6 shows the comparison between our hierarchical ciphertext storage scheme and other typical methods. Our tool can be integrated in the user client in the form of SDK, won't effect user experience like PGP. The user is in full control of his own key, without worrying about integrity of the agent bringed by CASB. Further, it combined with searchable encryption, won't affect data availability like MessageGuard. Moreover, we management behavior data based on blockchain, which makes the scheme traceable, thus improves security.

To sum up, in terms of user management and data authorization, the key management center only needs to issue the root key to users, and users can derive the sub-encryption key by himself, which reduces the cost of key management, it realize flexible secure key and user authority management at the file block level. It provides technical safeguards for clarifying the ownership of files, data, identifying security responsibilities, supervising the use of files and data. It relieves the worries of document providers, document demanders and managers, and creates safe convenient conditions for all units and departments to actively participate in data sharing and exchange.

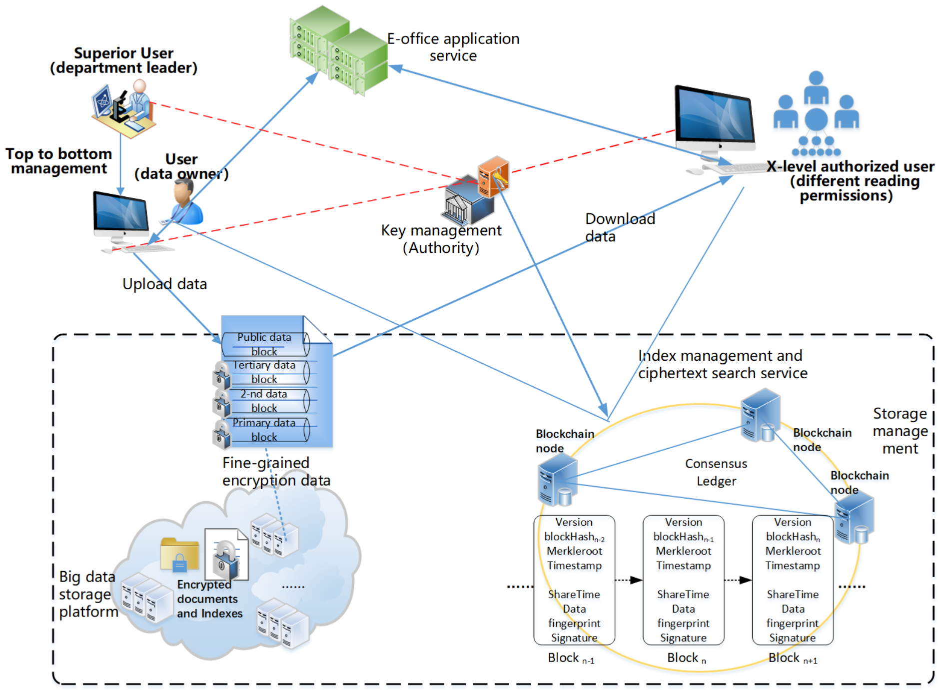

Fig. 13. Schematic diagram of data sharing application in company e-office.

Table 6. Comparison of related work.

Related work	Encrypted position in Fig. 2	Transparent encryption	Support general app	Key management position	Categories	Traceability
PGP [9]	A(Client)	×	√	User	File encryption tool	×
Message-Guard [11]	A(Browser)	√	√	Key management server	secure middle layer	×
CASB [18, 19]	B(Gateway)	√	×	Gateway	Cloud Access Security Broker, gateway	×
CryptDB [20]	C(Agent server)	√	×	Agent Server	Trusted agent	×
BCS(Ours)	position between A and B (Client/ Browser)	√	√	User and Key management server	Encryption tool(SDK)	√

5. CONCLUSIONS

In this paper, we realize blockchain-based hierarchical access control in terms of encrypted storage by classifying users, distributing corresponding master keys and auth-level keys, the keys of file contents are derived from users' master keys, file unique identifiers and data lables. Hierarchical and fine-grained control is realized through hierarchical keys, thus, users of corresponding levels can complete file encryption, upload and sharing only by holding encryption keys of corresponding levels, which is simple and convenient, effectively solve the content security problem caused by traditional simple whole encrypted storage, that is, the risk of large-scale data leakage caused by excessive authorization and single key leakage in data flow. Further, we use blockchain as a means of trust enhancement and backup, greatly balanced security and efficiency. We analyze and evaluate the scheme, and the application results show the efficiency in terms of authority division and security. As far as we know, we are the first to propose top-down key management and block-level fine-grained precise encryption authorization scheme to support e-government and e-office.

In our future work, we will continue to research the ciphertext computing in a ciphertext retrieval system on the basis of ensuring efficiency and continue to pay attention to the empowerment of blockchain to data security. We will also study the secure multi-party computation in cloud-edge collaborative distributed storage.

ACKNOWLEDGMENT

This work is supported by the Sichuan Science and Technology Program (2021 JDRC0077), the Sichuan Province's Key Research and Development Plan "Distributed Secure Storage Technology for Massive Sensitive Data" Project (2020YFG0298), and Applied Basic Research Project of Sichuan Province (No. 2018JY0370).

REFERENCES

1. Commerce Department Prohibits, "WeChat and TikTok transactions to protect the national security of the United States," https://2017-2021.commerce.gov/news/press-releases/2020/09/commerce-department-prohibits-wechat-and-tiktok-transactions-protect.html, 2020.
2. One of the investigation reports of the NSA cyber attack on NPU, http://www.chinadaily.com.cn/a/202209/06/WS631691c6a310fd2b29e761a2.html, 2022.
3. Didi was fined over 1 billion for violating laws, http://www.chinadaily.com.cn/a/202207/21/WS62d8e932a310fd2b29e6da1d.html, 2022.
4. Safe cow, large-scale face-beating scene, Ultra-large-scale data leakage in the world's safest cloud backup, https://aqniu.com/news-view/66379.html, 2020.
5. InfoQ, "The data leakage of year 2019 was shocking," https://cloud.tencent.com/developer/news/561392, 2020.
6. Cyber Risk Analytics (CRA) and Risk Based Security (RBS), "2021 data breach quick view report," https://pages.riskbasedsecurity.com/hubfs/Reports/2021, 2021.

7. A Study Conducted by Ponemon Institute and Sponsored, Analyzed, Reported by IBM Security, "2021 cost of a data breach report," https://branden.biz/wp-content/uploads/2021/08/Cost-of-af-DATA-Breach-Report-2021.pdf, 2021.
8. Alarming Cyber Security Facts to Know for 2021 and Beyond, https://www.cyber-talk.org/2021/12/02/alarming-cyber-security-facts-to-know-for-2021-and-beyond/, 2021.
9. Symantec, "Symantec desktop email encryption end-to-end email encryption software for laptops and desktops," http://www.symantec.com/desktop-email-encryption, 2016.
10. S. Ruoti, K. Seamons, and D. Zappala, "Layering security at global control points to secure unmodified software," in *Proceedings of IEEE International Conference on Cybersecurity Development*, 2017, pp. 42-49.
11. S. Ruoti, J. Andersen, T. Monson, *et al*., "MessageGuard: A browser-based platform for usable, content-based encryption research," *arXiv Preprint*, 2016, arXiv:1510.08943.
12. Terms V C. Virtru: Email encryption and data security for business privacy, https://www.virtu.com, 2019.
13. W. He, D. Akhawe, *et al*., "ShadowCrypt: Encrypted web applications for everyone," in *Proceedings of ACM SIGSAC Conference on Computer and Communication Security*, 2014, pp. 1028-1039.
14. X. Guo, Y. Huang, J. Ye, *et al*., "ShadowFPE: New encrypted web application solution based on shadow DOM," *Mobile Networks and Applications*, 2020, pp. 1-14.
15. B. Lau, S. P. Chung, C. Song, *et al*., "Mimesis Aegis: A mimicry privacy shield-A system's approach to data privacy on public cloud," *Georgia Institute of Technology*, 2014, pp. 33-48.
16. Gartne, "Market guide for cloud access security brokers," https://www.gratner.com/document/3488119, 2016.
17. Gartner, "Top 10 security projects for 2019," https://www.gratner.com/en/documents/3900996/top-10-security-projects-for-2019, 2019.
18. NETWORK S. Skyhighnetworks, https://www.skyhighnetworks.com/, 2019.
19. Ciphercloud, https://www.ciphercloud.com/, 2019.
20. R. A. Popa, C. M. S. Redfield, N. Zeldovich, *et al*., "CryptDB: Protecting confidentiality with encrypted query processing," in *Proceedings of ACM Symposium on Operating Systems Principles*, 2011, pp. 85-100.
21. R. Poddar, T. Boelter, and R. A. Popa, "ARX: an encrypted database using semantically secure encryption," in *Proceedings of the VLDB Endowment*, Vol. 12, 2019, pp. 1664-1678.
22. R. A. Popa, E. Stark, *et al*., "Building web applications on top of encrypted data using Mylar," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation*, 2014, pp. 157-172.
23. E. H. Beni, B. Lagaisse, W. Joosen, *et al*., "DataBlinder: A distributed data protection middleware supporting search and computation on encrypted data," in *Proceedings of the 20th International Middleware Conference Industrial Track*, 2019, pp. 50-57.
24. S. Belguith, N. Kaaniche, M. Laurent, *et al*., "Phoabe: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted IOT," *Computer Networks*, Vol. 133, 2018, pp. 141-156.

25. G. Chen, T. H. Lai, M. K. Reiter, *et al*., "Differentially private access patterns for searchable symmetric encryption," in *Proceedings of IEEE Conference on Computer Communications*, 2018, pp. 810-818.
26. N. Satoshi, "Bitcoin: A peer-to-peer electronic cash system," Satoshi Nakamoto Institute, https://nakamotoinstitute.org/bitcoin/, 2008.
27. M. Hao, *et al*., "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, Vol. 16, 2019, pp. 6532-6542.
28. F. Kai, *et al*., "A Blockchain-based clock synchronization scheme in IoT," *Future Generation Computer Systems*, Vol. 101, 2019, pp. 524-533.
29. Hyperledger, https://www.hyperledger.org/, 2020.
30. W. Jiang, H. Li, G. Xu, M. Wen, G. Don, X. Lin, "PTAS: Privacy-preserving thin-client authentication scheme in blockchain-based PKI," *Future Generation Computer Systems*, Vol. 96, 2019, pp. 185-195.
31. H. He, L. Zheng, P. Li, L. Deng, L. Huang, X. Chen, "An efficient attribute- based hierarchical data access control scheme in cloud computing," *Human-Centric Computing and Information Sciences*, Vol. 10, 2020, pp. 1-19.
32. S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, "TrustAccess: A trustworthy secure cipher-text-policy and attribute hiding access control scheme based on blockchain," *IEEE Transactions on Vehicular Technology*, Vol. 69, 2020, pp. 5784-5798.
33. 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, 2020, pp. 59389-59401.
34. X. Qin, Y. Huang, Z. Yang, and X. Li, "A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing," *Journal of Systems Architecture*, Vol. 112, 2021, p. 101854.
35. Y. Zuo, Z. Kang, J. Xu, and Z. Chen, "BCAS: A blockchain-based ciphertext-policy attribute-based encryption scheme for cloud data security sharing," *International Journal of Distributed Sensor Networks*, Vol. 17, 2021, No. 1550147721999616.
36. S. Athanere and R. Thakur, "Blockchain based hierarchical semi-decentralized approach using IPFS for secure and efficient data sharing," *Journal of King Saud University – Computer and Information Sciences*, Vol. 34, 2022, pp. 1523-1534.
37. Q. Zhang and Z. Zhao, "Distributed storage scheme for encryption speech data based on blockchain and IPFS," *Journal of Supercomputing*, Vol. 79, 2023, pp. 897-923.
38. F. Sammy, S. Vigila, *et al*., "An efficient blockchain based data access with modified hierarchical attribute access structure with CP-ABE using ECC scheme for patient health record," *Security and Communication Networks*, Vol. 2022, 2022, No. 8685273.
39. NIST, https://csrc.nist.gov/publications/, 2022.
40. Q. Xia, E. B. Sifah, *et al*., "MeDShare: Trust-less medical data sharing via blockchain," *IEEE Access*, Vol. 5, 2017, pp. 14757-14767.
41. F. Kai, W. Shangyang, *et al*., "MedBlock: Efficient and secure medical data sharing via blockchain," *Journal of Medical Systems*, Vol. 42, 2018, pp. 1-11.
42. D. X. Song, D. Wagner, *et al*., "Practical techniques for searches on encrypted data," in *Proceedings of IEEE Symposium on Security and Privacy*, 2000, pp. 44-55.

43. W. Jianfeng, C. Xiaofeng, and S. Shifeng, *et al*., "Towards efficient verifiable conjunctive keyword search forlarge encrypted database," in *Proceedings of the 23rd European Symposium on Research in Computer Security on Computer Security*, 2018, pp. 83-100.
44. S. Shifeng, J. K. Liu, A. Sakzad, *et al*., "An efficient non-interactive multi-client searchable encryption with supportfor boolean queries," in *Proceedings of the 21st European Symposium on Computer Security*, 2016, pp. 154-172.
45. W. Yunling, W. Jianfeng, S. Shifeng, *et al*., "Towards multi-user searchable encryption supporting boolean query and fast decryption," *The Journal of Universal Computer Science*, Vol. 23, 2019, pp. 222-244.

Yuxiang Chen received the BS degree from the University of Electronic Science Technology of China, China in 2016, received the MS degree from China Academic of Electronics and Information Technology, China in 2019, and he is currently pursuing the Ph.D. with the School of Computer Science and Engineering. His research interests include cryptography, cloud storage, and blockchain technology.

Yao Hao is a Senior Engineer of Science and Technology on Communication Security Laboratory, China and China Electronics Technology Cyber Security Co., Ltd. His research interests include applied cryptography, big data security and secure storage.

Zhongqiang Yi is a Senior Engineer of Science and Technology on Communication Security Laboratory, China and China Electronics Technology Cyber Security Co., Ltd. His research interests include applied cryptography, big data security and secure storage.

Xiaoyu Guo is an Engineer of Science and Technology on Communication Security Laboratory, China and China Electronics Technology Cyber Security Co., Ltd. His research interests include applied cryptography, and information security.

Chunxiang Xu received the BS, MS, and Ph.D. degrees from Xidian University, China, in 1985, 1988, and 2004, respectively. She is currently a Professor at the University of Electronic Science Technology of China, where she is involved in information security, cloud computing security, and cryptography.