CROSS: Supervised Sharing of Private Data over Blockchains

Mingxin Yin[1], Jiqiang Gao[1], Xiaojie Guo[1], Mengyao Sun[1], Zheli Liu[1(✉)], Jianzhong Zhang[1], and Jing Wang[2]

[1] College of Cyber Science, Nankai University, Tianjin, China
{yinmx0315,jiqiang,xiaojie.guo,mysun}@mail.nankai.edu.cn,
{liuzheli,zhangjz}@nankai.edu.cn
[2] Bubi Technologies Ltd, Beijing, China
wangjing@bubi.cn

Abstract. The transparent property of the blockchain guarantees the immutability of the data on the chain, but it can lead to violations of data privacy protection. On the other hand, absolute anonymity will make it difficult for the government to supervise the encrypted content stored on the chain. Moreover, it is inconvenient for the data owners to delegate their decryption authority to others. In order to solve the problem of data privacy concern and supervision in the current blockchain, we propose a supervised data sharing model called CROSS, which combines the proxy re-encryption mechanism with the tree key distribution mechanism. The model realizes the hierarchical supervision and horizental sharing of private data on the blockchain, which effectively improves the privacy of the blockchain while taking into account security. Consideration should be given to some potential attacks and corresponding defenses against our proposed model.

Keywords: Blockchian · Proxy re-encryption · Hierarchical supervision · Data sharing

1 Introduction

Blockchain technology has broken the centralization nature of the traditional Internet, making the crisis of confidence to plague the modern economy resolved to some extent. The transparency is one of the important features of blockchain (i.e., the data stored on the blockchain is visible to any node), but it is a double-edged sword. It ensures the behavior that attempts to tamper with the data is recorded and discovered, but in turn, the privacy of data is at risk. Users don't want anyone to view their data at any time, especially in the business or government affairs. The data of governments or companies, etc. needs to be kept private strictly. Meanwhile, the transfer of decryption rights to a particular user may occur in a particular scenario. How to securely share private data on the blockchain is a challenge.

J. Li et al. (Eds.): SPNCE 2019, LNICST 284, pp. 73–86, 2019.
https://doi.org/10.1007/978-3-030-21373-2_7

A simple solution is to encrypt the data before it is released to the chain, but this makes it harder for the authorities to supervise the data. Government departments or regulators should be able to conduct regulatory review of company data to ensure that data retained or shared on the chain is in compliance with legal requirements. For encrypted data, regulators need a feasible regulatory scheme.

Existing blockchain-based data sharing platforms, such as Blockstack [1] and Calypso [2], are facing the problems above. A feasible and effective solution should meet the following requirements simultaneously:

Hierarchical Supervision. When a new node joins, the authority assigns a key to it based on its hierarchy. The authoritative node can calculate the key of the lower-level node, and view the encrypted data that is stored and shared by the subordinate nodes without authorization, so as to ensure the security of the data.

Private Data Sharing. Nodes can maintain fine-grained access control for data on the chain. Other users (non-authoritative nodes) can obtain the partial data after having the data owner's authorization. In this process, the plaintext will not be leaked to any third party other than the data owner and authorized user.

Our Contribution. The contributions of this paper are summarized as below:

Tree-Based Key Distribution Mechanism. We design a key distribution mechanism based on tree structure, which can effectively realize hierarchical membership of key. The authoritative node can calculate the key of the subordinate node, but the subordinate node cannot reversely crack the key of the authoritative node.

Supervised Private Data Sharing Model. We propose a supervised private data sharing model based on the consortium blockchain, combined with proxy re-encryption and key tree (hierarchical key distribution) scheme, that has realized hierarchical supervision and sharing of the encrypted data on the chain, effectively improves the privacy of blockchain while taking into account the security. Some potential attacks which can attempt on the proposed model and how the model can handle such attacks are also discussed.

2 Related Work

There have been various attempts to address the problems of data privacy, from both the system level and specific technology level. As an emerging technology, blockchain plays an important role in the field of privacy protection.

Data Management Systems Based on Blockchain. Some data management systems [3,5] based on blockchain have been proposed, their starting point

is to guarantee the security of access control management and log audit by utilizing the immutability of blockchain. Chain Anchor [5] attempts to provide anonymous but verifiable identities for entities, which seek to submit transactions to the blockchain. Zero knowledge proof (ZKP) is the core stone for5 proving the anonymous membership in it.

Data Sharing System Based on Blockchain. A large part of the research on the privacy data sharing system of blockchain is applied to the medical data sharing scenario. BBDS [4] is a data sharing framework based on consortium blockchain, which uses encryption operations (e.g., encryption and digital signature) to ensure the effective access control for sensitive shared data pools. MedRec [7] and MedShare [8] are generally similar, both of them control the operations of exchanging data between the providers' databases via smart contract. Xu [10] proposed the blockchain combined with homomorphic encryption to build a scheme for protecting privacy of electronic healthy record. Proxy re-encryption is a novel method for designing secret sharing schemes [2,6,9], that allows third party to alter a ciphertext which has been encrypted for one party, so that it may be decrypted by another. CALYPSO [2] is a decentralized and auditable blockchain framework for data sharing.

In this paper, in addition to considering private data sharing, we also provide solutions of hierarchical supervision for encrypted data on the chain.

3 Preliminaries

Our resarch is on blockchain, combined with proxy re-encryption which is implemented based on bilinear mapping. In this section, we review some basic cryptology concepts which will be needed later in this paper.

3.1 Blockchain

Blockchain is a distributed ledger technology that maintains a continuously growing list of records in a verifiable and permanent way by cryptography. Multiple parties can jointly maintain the ledger through a specific consensus mechanism, so that the data on the chain has the characteristics of decentralization and strong consistency. Since the blockchain has the feature of immutability, it can be the underlying architecture of many trusted distributed systems, not just in the field of cryptocurrency [14,15]. In the process of landing, the blockchain technology often encounters resistances due to the difficulties of regulation and privacy issues.

Based on the consortium blockchain, we implement a supervised private data sharing model. The blockchain is used in the model as a publicly verifiable and chronologically-ordered immutable database. Each node in the consortium encrypts part of its own public data, and broadcasts the digital digest and additional information to the entire network. All nodes record and update the data operation on the local ledger, that makes adversary difficult to tamper with the data and avoids the single point of failure. Each block of the chain is composed

of transactions and each transaction records a data processing event. All nodes follow the Longest Chain Rule [14] to maintain the uniqueness of ledger.

In addition, compared with the traditional node key generation mechanism, we propose a tree-based key distribution supposed hierarchical supervision. The private key of each node is no longer randomly generated, but is distributed by the root node in the consortium according to the key generation policy. The node's private key participates in the data encryption process, so that the authoritative node can realize the supervision.

3.2 Bilinear Groups

Let $\mathcal{G}$ setup be a bilinear group generator that on input the security parameter, outputs the parameters for a bilinear map as $(q, g_1, g_2, G_1, G_2, G_T, e)$, where G_1, G_2 (Here, G_1 may equal G_2) and G_T are groups of prime order q and g_1, g_2 are generators of G_1, G_2, respectively. The bilinear map e: $G_1 \times G_2 \rightarrow G_T$ is efficiently computable and non-degenerate.

In the proxy re-encryption protocol, we use bilinear mapping to improve the security of the key, and also resist the occurrence of collusion attacks by the proxy and the receiver.

3.3 Proxy Re-Encryption

Proxy re-encryption is a ciphertext conversion mechanism under public key cryptography(PKC). It was first proposed by Blaze et al. in 1998 [11], subsequently some improved schemes were proposed by [12,13]. Proxy re-encryption mechanism converts the ciphertext encrypted with data owner's public key into ciphertext encrypted with data user's public key and no leaks of private key or plaintext. Several main functions are as follows:

Init(). User initialization generates public and private keys. The input is as the security parameters 1^k, key generator generates (pk_A, sk_A), (pk_B, sk_B).

Encrypt(pk_A, m***).*** Encrypt plaintext m with pk_A, return ciphertext c.

RekeyGen(sk_A, pk_B***).*** Generate a re-encryption key according to A's private key and B's public key, return $rk_{A \rightarrow B}$.

Re-encrypt($rk_{A \rightarrow B}$, c***).*** Use the $rk_{A \rightarrow B}$ to perform ciphertext conversion and return the re-encrypted ciphertext Rc.

Decrypt(Rc, sk_B***).*** Decrypt the Rc with sk_B, return the plaintext m.

For example, the data owner Alice wants to share the data encrypted by her public key with Bob relying on the third-party proxy without leaking the private key and plaintext. The working process is as follows:

Alice and Bob call the function $init()$ to get their own public key and private key respectively. Alice uses the public key pk_A to encrypt the plaintext m to ciphertext c and stores c in cloud database. Then Alice combines part of her private key with part of Bob's public key to generate the proxy re-encryption key $rk_{A \rightarrow B}$, which is sent to the proxy. The proxy re-encrypts the ciphertext c to Rc using $rk_{A \rightarrow B}$, then sends Rc to Bob. Bob decrypts the Rc with his own private key sk_B.

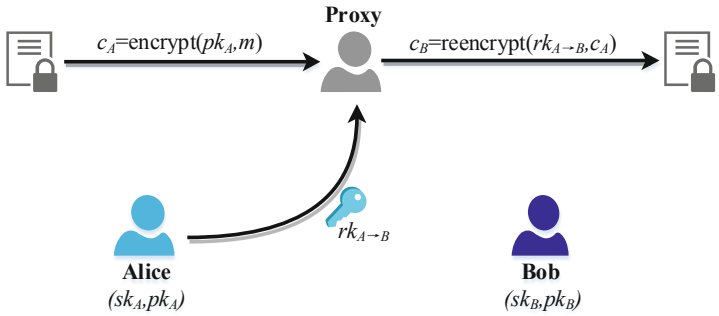

Fig. 1. The interaction process of proxy re-encryption

Proxy re-encryption has been used in many interesting applications, such as mail forwarding system [12], distributed file system [12] and so on. In our model, proxy re-encryption as a core stone in the data sharing scenario, can realize the trusted sharing of private data on the blockchain (Fig. 1).

4 System Model

The proposed model aims to solve the problem of private data sharing and hierarchical supervision simultaneously. In this section we present an overview of CROSS, then describe the model sub-module in detail.

4.1 Overview

Goals. The goals of designing the model are as follows:

Privacy-Persevering. In the process of private data sharing, the data will not be disclosed to any organizations or individuals other than the data owner and the authorized data user. The proxy re-encrypts the ciphertext, but it can not touch the plaintext;

Supervised. The authoritative node can calculate the key of the lower-level node according to the key generation policy, and can view the encrypted data at any time without informing the subordinate node, so as to realize data supervision;

Auditable. The data owner encrypts the data and publishes the digital digest to the chain, ensuring that the data is not tamperable. Simply, if the new digital digest is not the same as the one on chain, it indicates that the data has been tampered. The blockchain provides an effective way to acheive audit.

User Roles. There are five kinds of participating roles in the model, namely Membership Service Provider(MSP), data owner, data user, proxy server and authoritative node.

MSP is a consortium blockchain member management module containing a set of trusted nodes, each node holds a key pair, the keys of all nodes in the MSP are combined to form the root key *kroot* of the key tree, $k_{root} = (k_{r_1} \oplus k_{r_2} \oplus k_{r_3} \oplus \cdots \oplus k_{r_n}) \bmod p$. MSP is the starting node and responsible for issuing nodes' private keys that are used for encrypting data.

Data owner refers to the node that published private data to the chain. Data owner has a pair of public key and private key. A random generated key is used for encrypting data and the data owner uses his private key to encrypt the random key. When data owner needs to share data with others, he authorizes the other nodes by generating re-encryption key so that the data user can decrypt data with his own private key.

Data user refers to the node that wants to get data from data owner and has been authorized by data owner. Data user requests proxy server to re-encrypt ciphertext, obtains the converted ciphertext and decrypts it using his private key. There is no limit to the hierarchical relationship between data user and data owner.

Proxy server refers to the server that converts the ciphertext encrypted by the data owner into a format that data user can decrypt when the data owner shares data with the data user. The data owner sends the re-encryption key to the proxy server, and the data user requests the proxy server to perform the re-encryption operation. Suppose the proxy server is semi-trusted, that is to say, the server performs this process accurately, but it will always try to understand the ciphertext.

Authoritative node is the relative concept of data owner, refering to the high-level node of the data owner. Authoritative node can calculate data owner's private key according to the key generation policy and view the encrypted data without authorization of the subordinate node.

Collusion Attack. The proxy re-encryption scheme in model resists collusion attack. In common BBS98 scheme [11], the proxy server and data user may collude to crack the data owner's private key, such as the proxy server has the re-encryption key $a^{-1}b$ and the data user provides b^{-1}, they can crack the data owner's private key a by simply calculating the inverse of the result a^{-1} that is from multiplying $a^{-1}b$ and b^{-1}. The proxy re-encryption scheme used in our model is based on bilinear group, that can defend against such attacks.

Malicious Access. Suppose a malicious user wants to write and read encrypted data. For write operation, data can be only written to the blockchain after the signature has been successfully verified. For read operation, there are only two conditions: (1) the authoritative node reads its subordinate node's data. (2) only the authorized user can read the re-encrypted ciphertext.

4.2 System Design

The tasks we completed with the CROSS model are as follows: First, the CROSS controls two processes, the registration and the key distribution. Then, the high-level authoritative node supervises the data encrypted by the lower node, and

the nodes can share private data via proxy utilizing the re-encryption mechanism. Finally, for some invalid nodes, undo and reclaim their keys. It is mainly divided into three modules, namely key management module, data sharing module and data supervision module. Generally, the model can be described as $CROSS = \{keyManagement, dataSharing, dataSupervision\}$. In particular, the key management module includes two cases: add node and undo node. The overall architecture of the model is shown in Fig. 2.

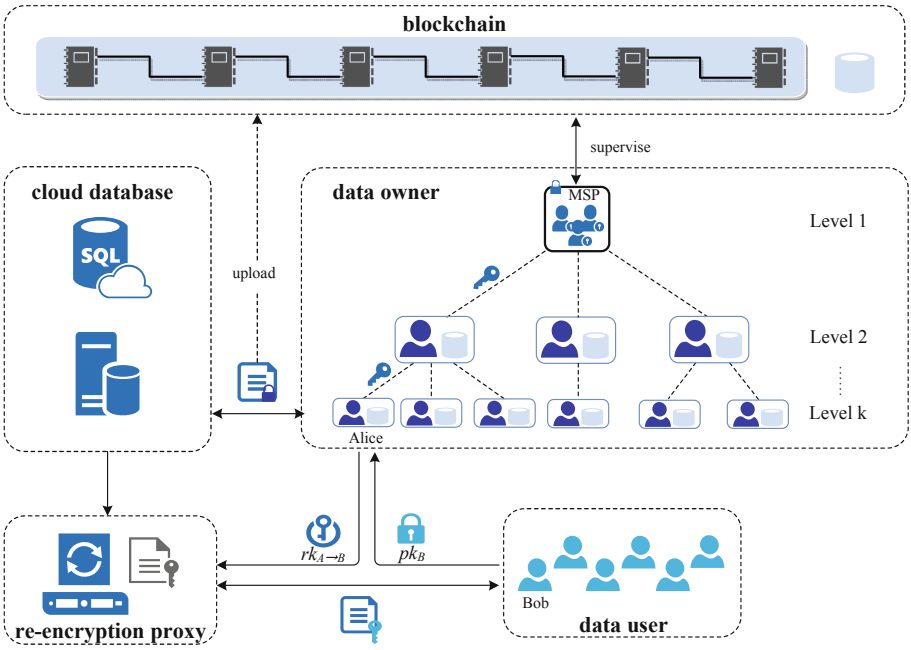

Fig. 2. The overall architecture of CROSS

Threat Model. We considered two attacks, collusion attacks and malicious access attacks.

Key Management Module. We have designed a key distribution mechanism based on tree structure, which effectively realizes the hierarchical relation of key management, as shown in Fig. 3. The key of each node is related to its parent node's key, and the path from the root node to each node is taken as one of the hash factors to generate keys to reduce the probability of repetition between two node's key. In order to achieve the security of the undo mechanism, the undo state value S is also introduced as one of the hash factors. When the key tree is established, the root key *kroot* is set for the root node of the key hash tree, and the revocation state value S of all nodes is set to 0, indicating that it has never been revoked.

Add Node. Whenever a new node joins, its parent node requests the root node to distribute a key, and the root node calculates the key value of the new node according to the parent node key value, path value and state value. That is, the key derivation policy is: $k_{new} = H(k_{parent} \parallel path_{new} \parallel S_{new}) \bmod p$, among them, k_{parent} is the key corresponding to the new node's parent node, which is known to root node. $Path_{new}$ is the path from the root node to the new node. S_{new} represents the state value. $H()$ is the SHA-256 digest algorithm, $\parallel$ represents the concatenation operation, and p is a prime number. In Fig. 3, we suppose that node N is willing to join, and its parent node is node E. Node E requests the root node to distribute the key for node N. The root node determines the key of the new node as $k_N = H(k_E \parallel 010 \parallel 0) \bmod p$ according to the key distribution policy.

Undo Node. When the user performs key undo operation on some nodes, the key revocation state value S of the corresponding node is increased by 1, $S = S+1$, indicating that the node has been revoked once. The MSP recalculates the new key according to the key generation policy formula, assigns it to the new user who takes over the location of the node that was revoked. Note that the keys of all the subordinate nodes of the revoked node need to be recalculated. In this way, the old key of the original user is invalidated, ensuring the security of the new user's data.

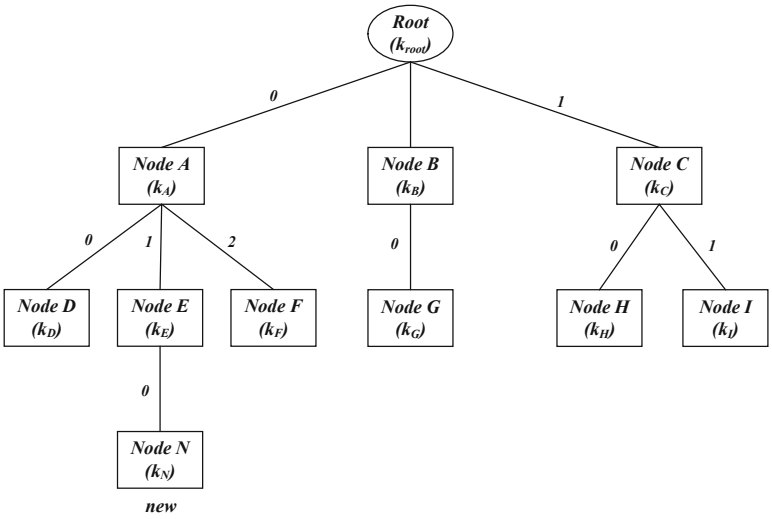

Fig. 3. The structure of key derived tree

According to the principle of asymmetric encryption, we generate public key and private key to each node in the consortium blockchain. In the previous section, we mentioned that the model uses the proxy re-encryption scheme based on bilinear mapping to improve the security of the key. The derived tree

distributes the private key for each node, and the public key is generated according to the private key. So for node N, the private key of its parent node is $sk_E = (e_1, e_2)$, the private key of node N is $sk_N = (n_1, n_2)$,

$$n_1 = H(e_1 \parallel 010 \parallel 0) \bmod p,$$

$$n_2 = H(e_2 \parallel 010 \parallel 0) \bmod p,$$

the public key of node N is $pk_N = (Z^{n_1}, g^{n_2})$.

Data Sharing Module. In the data sharing module, there are three kinds of entities: data owner, data user and proxy server. The interaction process is shown in Fig. 4. In our scheme, the data set $DST = (metadata, edek, epos, hash(m))$ is published to the chain. The real ciphertext is stored in cloud databases.

In each round of sharing, the data owner always does this first: Generates a random key $dek = random()$ ($dek \in G_T$) for encrypting data, and uses the random key dek to encrypt plaintext m to ciphertext c, $c = encrypt_{sym}(dek, m)$. Next, the data owner stores the ciphertext c in the cloud database and returns the storage location pos, subsequently encrypts (dek, pos) with the public key to $(edek, epos)$, signs $DST = (metadata, edek, epos, hash(m))$ and then publishes it to the chain.

We take Alice as data owner and Bob as data user to illustrate the process of data sharing in Fig. 4. Alice and Bob already have $sk_A = (a_1, a_2)$, $pk_A = (Z^{a_1}, g^{a_2})$ and $sk_B = (b_1, b_2)$, $pk_B = (Z^{b_1}, g^{b_2})$. When Alice wants to share data with Bob, Alice generates the re-encryption key $rk_{A\to B} = rekey(sk_A, pk_B)$ based on part of her private key a_1 and part of Bob's public key g^{b_2},

$$rk_{A\to B} = (g^{b_2 a_1}), \tag{1}$$

and sends the re-encryption key to the proxy server. Note that every public key is visible to all nodes.

After the proxy server received $rk_{A\to B}$ and saved it. Bob gets $(edek, epos)$ from the chain and requests the proxy server to convert the encrypted data. In response, the proxy server accepts the re-encryption request from Bob and converts $(edek, epos)$ that needs to be decrypted with sk_A into $(edek', epos')$ that can be decrypted with sk_B,

$$(edek', epos') = re - encrypt(rk_{A\to B}, (edek, epos)). \tag{2}$$

According to the algorithm in [12], the ciphertext has two types:

(1) First-level Encryption E_1. The ciphertext can be only decrypted by Z^{a_1},

$$E_1(m, pk_A) = (Z^{a_1 k}, mZ^k). \tag{3}$$

(2) Second-level Encryption E_2. The ciphertext is used to do the proxy re-encryption operation,

$$E_2(m, pk_A) = (g^k, mZ^{a_1 k}). \tag{4}$$

The proxy re-encryption process can be understood as converting the second-level ciphertext of Alice to the first-level ciphertext of Bob. The plaintext need to be encrypted is (dek, pos). Input the $rk_{A\to B} = g^{a_1 b_2}$ and Alice's Second-level ciphertext $edek = (g^k, dekZ^{a_1 k})$, $epos = (g^k, posZ^{a_1 k})$. According to the properties of bilinear mappings,

$$e(g^k, g^{a_1 b_2}) = e(g, g)^{b_2 a_1 k} = Z^{b_2 a_1 k}. \tag{5}$$

The result after the re-encryption conversion is

$$(Z^{b_2 a_1 k}, dekZ^{a_1 k}) = (Z^{b_2 k'}, dekZ^{k'}) = edek', (k' = a_1 k), \tag{6}$$

similarly, $epos'$ can be obtained. At this point, $edek'$ and $epos'$ are the first-level ciphertext that can only be decrypted with sk_B. Bob decrypts and gets the plaintext,

$$dek = dekZ^{k'} \cdot ((Z^{b_2 k'})^{b_2^{-1}})^{-1} = dek \cdot Z^{k'} \cdot (Z^{k'})^{-1} = dek(\mathrm{mod} p). \tag{7}$$

The pos is calculated in the same way. Bob gets the dek, which can be used to decrypt the data at the pos position.

Then proxy server sends $(edek's, epos's)$ to Bob for his decryption, Bob decrypts $(edek', epos')$ with sk_B,

$$(dek, pos) = decrypt(sk_B, (edek^{'}, epos^{'})). \tag{8}$$

The dek can be used to decrypt the data in the cloud database located in the pos, and finally Bob generates a digital digest of the decrypted plaintext and verifies whether the data is tampered or not by comparing with the existing digital digest on the chain.

Data Supervision Module. The model is designed in the consortium blockchain scenario, the private key of the authoritative node determines the key of the subordinate node, and the authoritative node can have the permission of decrypting data encrypted by the subordinate node, but the ciphertext generated by the authoritative node cannot be decrypted by the subordinate node, in this way, while ensuring the anonymity of data, the content can be regulated and the availability of blockchain can be extended. In the data supervision module, it includes two entities: data owner and data user. The interaction process is shown in Fig. 5.

We take Alice as the data owner to illustrate the process of data supervision. The authoritative node wants to view the encrypted data after Alice encrypts the data and publishes it to the chain. The simple way for authoritative node is to

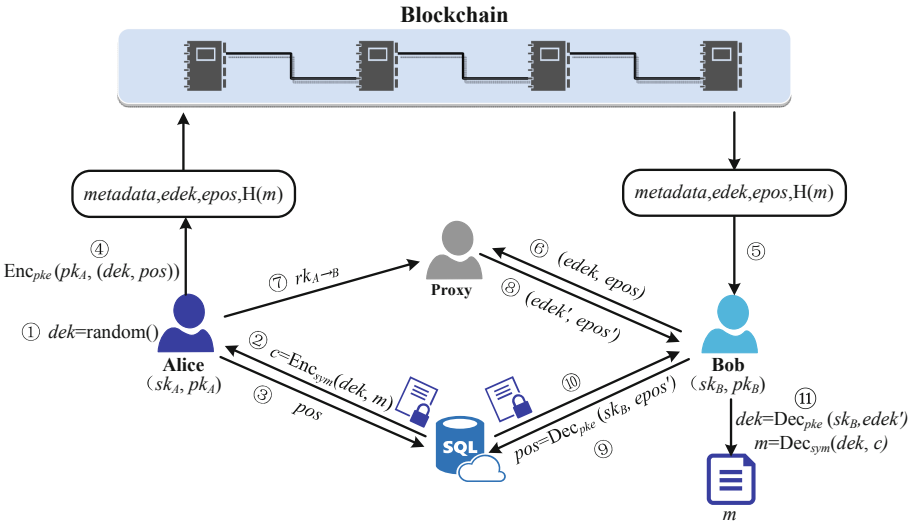

Fig. 4. The interaction process of privacy data sharing

calculate Alice's private key sk_A according to the key generation policy so that he can decrypt $(edek, epos)$ with sk_A to obtain (dek, pos). To obtain the key dek, the data in the cloud database located in the pos can be easily decrypted. Finally the authoritative node generates a digital digest of the decrypted plaintext, which can be verified whether the data is tampered by comparing with the digital digest on the chain.

5 Security Analysis

In order to meet the security of the CROSS, it is necessary to simultaneously satisfy the key management security, non-tampering of data and the invisibility of plaintext to proxy. In this section we analyze the security of the model. According to Theorems 1, 2 and 3, we prove that the proposed model is safe and feasible.

Theorem 1. *It is impossible for the lower node to crack the key of the authoritative node.*

Proof. In the model, the root node of the key distribution tree is composed of a set of trusted nodes. Each node holds a key, and the keys of all nodes are combined to form $kroot$, which ensures the security of the key distribution mechanism. Even if a single node in MSP is hijacked, the key of a subordinate node cannot be stolen. The key of each node in the tree is associated with the superior node. According to the characteristics of the hash function, it is easy for the authoritative node to calculate the key of its subordinate node, but the reverse is non-computable, which satisfies the requirements of supervision.

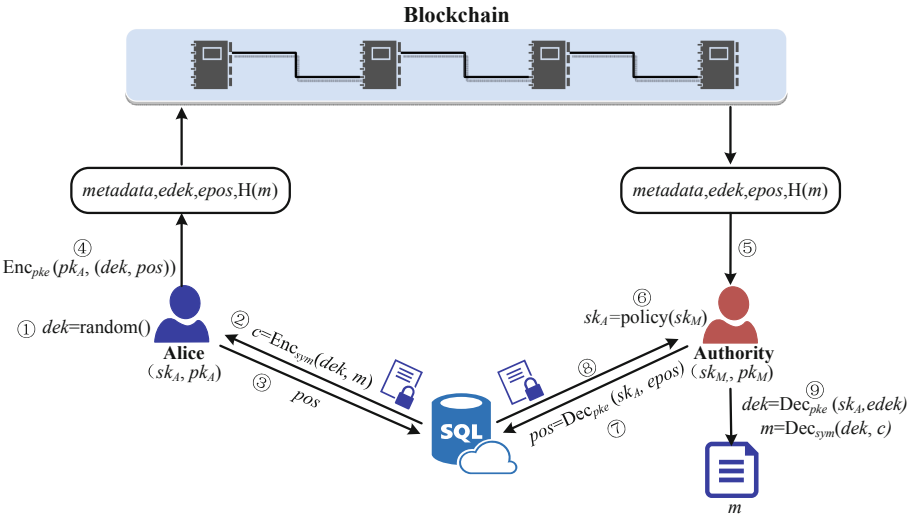

Fig. 5. The interactive process of data supervision

In addition, a revoked user can no longer read any data from the subordinate node with the original private key, after the user has been revoked, the revocation status value changes, the new node that replaces the revoked node will get the new key, and the private keys of all cascaded subordinate nodes will be updated, ensuring that only valid users can get the data.

Theorem 2. *The proxy server does not have access to plaintext.*

Proof. Data owners maintain an access control list that supports fine-grained access control of data and protects the privacy of data owners. Only the data owner can create a re-encryption key and set an access control policy, and the data owner can control other users' access to their data by updating the access list, that is, by adding or revoking the re-encryption key. In our model, it is assumed that the re-encryption proxy server is semi-trusted, that is to say, the server performs this process accurately, but it will always try to understand the data. Real data is encrypted using a random key *dek* generated by the dataowner, *dek* was encrypted to *edek* by the public key of data owner, the proxy server knows all the re-encryption keys, but it cannot get the real dek. Before performing re-encryption conversion, only the data owner can decrypt *edek*, after re-encryption conversion, only the authorized data user can decrypt the encrypted the $edek'$. The corresponding private keys are not revealed in the re-encryption process, thus the proxy server cannot access any plaintext data.

Theorem 3. *Malicious tampering with private shared data is impossible.*

Proof. The main advantage of blockchain is verified and non-repudiation. In our model, all access to data can be traced through the blockchain log. In order to

ensure the authenticity of the data, the signature is verified before the data is published to the blockchain. The authentication mechanism allows real users to access the data. On the blockchain, the digital digest of plaintext data is stored, after data user decrypts the ciphertext, he can generate a new digital digest of plaintext locally to compare with the digital digest on the chain, confirming whether the data has been tampered with or not.

6 Conclusions

Considering the problem of data privacy concern and supervision on the current blockchain, we propose a blockchain-based supervised private data sharing model, combined with proxy re-encryption scheme and key distribution mechanism based on tree structure. It realizes the hierarchical supervision and private data sharing on the chain, effectively improving the privacy of the blockchain. The security of the model is analyzed and proved in the paper. In future work, we will use the consortium blockchain to conduct experimental research on the proposed model and evaluate the experimental results to improve the model.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (No. 61672300), National Natural Science Foundation of Tianjin (No. 16JCYBJC15500), and Technology Research and Development Program of Tianjin (No. 18ZXZNGX00140). Jiqiang Gao has the same contribute to this paper with Mingxin Yin.

References

1. Ali, M., et al.: Blockstack: a global naming and storage system secured by blockchains. In: USENIX Annual Technical Conference, pp. 181–194 (2016)
2. Kokoris-Kogias, E., et al.: CALYPSO: Auditable Sharing of Private Data over Blockchains. Cryptology ePrint Archive 209 (2018)
3. Zyskind, G., Nathan, O.: Decentralizing privacy: using blockchain to protect personal data. In: Security and Privacy Workshops (SPW), pp. 180–184 (2015)
4. Xia, Q., Sifah, E.B., Smahi, A., et al.: BBDS: blockchain-based data sharing for electronic medical records in cloud environments. Information **8**(2), 44 (2017)
5. Hardjono, T., Pentland, A.S.: Verifiable Anonymous Identities and Access Control in Permissioned Blockchains (2016)
6. Cui, S., Asghar, M.R., Russello, G.: Towards blockchain-based scalable and trustworthy file sharing. In: International Conference on Computer Communication and Networks (ICCCN), pp. 1–2 (2018)
7. Azaria, A., Ekblaw, A., Vieira, T., et al.: Medrec: using blockchain for medical data access and permission management. In: Open and Big Data (OBD), pp. 25–30 (2016)
8. Xia, Q., Sifah, E.B., Asamoah, K.O., et al.: MeDShare: trust-less medical data sharing among cloud service providers via blockchain. IEEE Access **5**, 14757–14767 (2017)
9. Xue, T.-F., Fu, Q.-C., Wang, C., Wang, X.-Y.: A medical data sharing model via blockchain. Acta AutomaticaSinica **43**(9), 1555–1562 (2017)

10. Xu, W., Wu, L., Yan, Y.: Privacy-preserving scheme of electronic health records based on blockchain and homomorphic encryption. J. Comput. Res. Develop. **55**(10), 2233–2243 (2018)
11. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054122
12. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the 12th Annual Network and Distributed Systems Security Symposium (NDSS) (2005)
13. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_19
14. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). http://www.bitcoin.org/bitcoin.pdf
15. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Proj. Yellow Pap. **151**, 1–32 (2014)