A Data Access Control Method Based on Blockchain

JunWei Ma*, HongLin Xue, Feng Wang, Yi An, DongHui Han, Dong Wang, Min Zhao, and Sheng Bi

Information and Communication Branch, State Grid Shanxi Electric Power Company, No.3, Heyuan Road, Changfeng Business District, Jinyuan District, Taiyuan, China

*Email: MaJunWei@sx.sgcc.com.cn

Abstract. In the digital era, a large amount of data is generated every day and recorded on the storage system. Access control is an important mechanism for ensuring data privacy and security. Data accessing control model of traditional storage system is centralized, which has threatened the information security of users. As blockchain is emerging technology which is widely used in the area of data privacy preserving, in this paper we proposed a decentralized access control model via blockchain based smart contracts implementation. Once satisfying the conditions set by users, visitors could apply to the blockchain for authentication, then obtain permission to access user data and operate the data, finally achieving secure access control of user data. In our experiment, we build the blockchain system based on Hyperledger Fabric project and evaluate both the efficiency of writing and the verification of access permission. The result shows our model’s usability and efficiency.

1. Introduction

With the rapid development of information technology, a large number of data are generated every day and recorded in the storage system. Now distributed storage system is widely used in the field of data storage. Different from the traditional storage, it distributes data in multiple independent devices and provides users with burst and concurrent services. Because of the existence of sensitive information in the data, how to protect the sensitive information is a challenge to the design of modern storage system represented by distributed storage system. Access control is an important mechanism used to protect sensitive information in distributed system. It can prevent illegal unauthorized access by managing users' operation rights on different data. The access control mechanism maintains an access control list that records the user's access rights, and checks the user's operation requests. Traditional access control mechanism uses centralized access control list [1-4], this mechanism can be easy to suffer from single point of failure, which leads to the leakage and tampering of access right information, resulting in to security problems. At the same time, when file access rights become complex, using centralized access control to manage them will reduce their own efficiency.

As the technology behind Bitcoin [5], blockchain has attracted people's attention and research after its popularity, and has been applied in many fields [6-8]. Nowadays, blockchain system is considered to contain six parts [9]:

1) Data layer: blocks linked by cryptography, namely blockchain structure.
2) Network layer: peer-to-peer network nodes.
3) Consensus mechanism: used to synchronize the blockchain data of network nodes.
4) Smart contract: executable code used to implement behavioral logic.
5) Service layer: blockchain platform, such as Fabric, Ethereum.
(6) Application layer: specific blockchain applications, such as digital currency.

The blockchain has a decentralized network structure. All nodes in the network participate in data maintenance, so the data stored on the blockchain has the characteristics of tamper proof. At the same time, the consensus mechanism of blockchain ensures that all participating nodes store the same data. These characteristics of blockchain technology make it suitable for solving the single point of failure of access control settings in traditional storage systems.

Therefore, we propose a new privacy accessing control model based on the blockchain smart contract, named Access Control Contract (ACC). ACC enables the data owners in the distributed storage system to set the access rights of different users to their data. The distributed storage system will interact with ACC for data access rights when providing access services. ACC includes two modules: access rights configuration module and access rights verification module.

Access rights configuration is performed by the data owner. The data owner associates the data, the visitor's identity information and access rights, and records them on the blockchain through ACC. He can modify or delete the configured access information through ACC later.

Access rights verification is to verify the access rights of visitors in the data access phase. Visitors need to provide their own identity information. ACC verifies whether they have the right to access the specified data. If the access is allowed, the visitors can access and operate the data, so as to achieve privacy access control.

The advantages of our privacy access control model are as follows:

(1) The access control list is recorded on the blockchain, which solves the single point failure problem of traditional centralized storage.

(2) The access control list is separated from the storage service to solve the problem of low efficiency caused by complex access rights management.

The rest of this paper is arranged as follows. The second chapter will put forward the design scheme of access control contract; the third chapter will introduce the data access interface model of distributed system based on smart contracts; the fourth chapter will carry out experiments to verify the feasibility of our proposed scheme.

2. Smart Contracts

In order to realize the function of access control, we design an access control contract (ACC), which records the access rights of data on the blockchain, and provides the functions of query and subsequent modification. We will introduce the data structure of record access rights setting and the algorithm flow of ACC.

2.1. Three-Tier of Access Control

In order to achieve fine-grained access control, we consider that the data owner can set permissions for different data visitors, and the data owner may own multiple data, and the permission settings for these data can be different. Therefore, we use a three-tier table to represent the permission settings.

The first layer of the three-tier table is the property table, which is used to record the creation information, such as the creator and creation time. The second layer is the permission query table, which is used to record about the data owners have grant visitors any permission. The third layer is the permission configuration table, which is used to record the access permission settings corresponding to the specific data. Its structure is shown in Figure 1. Next, we will explain the specific contents of the three-tier table.
Three-Tier of Access Control.

The properties table records the creation information of the three-tier access table, and its contents are shown in table 1.

Table 1. Properties Table.

| ID | Lookup | GID     | Time         |
|----|--------|---------|--------------|
| A  | 0x000f | Group A | 2020-6-12 17:31 |
| ...| ...    | ...     | ...          |

As can be seen from table 1, the properties table contains the following information:

ID: the creator of the properties table, usually the data owner.
Lookup: the address of the permission query table. The permission query table records about the data owners have grant visitors any permission.
GID: group ID, users of the same group can modify the contents of these three tables.
Time: the creation time of this row in the properties table.

The contents of the permission query table are shown in table 2.

Table 2. Instance of permission query table.

| Did | Vid | AClist | time         |
|-----|-----|--------|--------------|
| A   | B   | 0xffff7788 | 2020-6-22 11:12 |
| C   | B   | 0xffff779a | 2020-6-22 12:45 |
| ... | ... | ...     | ...          |

From table 2, the permission query table lookup contains the following information:

Did: the identity of the data owner.
Vid: the identity of the data visitor.
AClist: address of permission setting table. The data permission table records the access permission setting of specific data.

The contents of the data permission table can be represented by table 3. In order to reduce the size of the data, it only records the permissions that are allowed, while the permissions that are not recorded are not allowed.

Table 3. Example of permission setting table.

| Data | permission | lasttime |
|------|------------|---------|
| File A | read   | 2020-5-13 16:18 |
| File A | write  | 2020-6-20 08:09 |
| ...   | ...     | ...     |
It can be seen from table 3 that data permission table contains the following information:
Data: used to identify the data resources that need access control.
Permission: used to identify the permission allowed on the corresponding data resource.
Last time: the last modification time of this data permission.

2.2. Access Control Contract

Access Control Contract (ACC) is used to create and maintain three-tier access rights table. It includes the following functions.
Create: used to create a new ACC.
Add: used to add a new access permission in the query table lookup.
Delete: used to delete specific existing access rights.
Check: used to query the user's access rights.

Here, we only provide the most basic functions. Many other functions can be logically combined by the above functions. For example, the update function can be combined by deleting and adding. For the convenience of algorithm description, we introduce the following symbols:
Create(): to create a new three-tier access control table;
Time(): returns the current system time;
CreateNewTable(): used to create a new lookup;
CreateNewAccList(): used to create a new acclist;
Row: the row indicating lookup or acclist;
MT: the three-tier access table.

The algorithm flow of ACC can be described by algorithm 1.

**Algorithm 1. ACCfun.**

```
Input: mT, Did,Gid, action, Vid, Data, permission
Output: output
1. Switch action:
2. Case: “create”:
3. mT=create();
4. mT{’id,’Gid’,’lasttime’,}={Did,Gid,time()};
5. ACC{’lookup’}=CreateNewtable();
6. Output=mT;
7. Case: “add”:
8. Row=mT{’lookup’}.find(’Did’:Did, ’Vid’:Vid);
9. If Row<0
10. mT{’lookup’}.append();
11. Row=ACC{’lookup’}.len;
12. mT{’lookup’}[Row]{’Did’,’Vid’}={Did,Vid};
13. mT{’lookup’}[Row]{’Acclist’}=CreateNewAcclist();
14. output=ACset(mT{’lookup’}[Row]{’Acclist’}, data, permission, ‘Add’)
15. Case: “delete”
16. Row=ACset{’lookup’}.find(’Did’:Did, ’Vid’:Vid);
17. If Row<0
18. output= 0
19. output=ACset(mT{’lookup’}[Row]{’Acclist’}, data, permission, ’delete’)
20. Case: ‘check’
21. Acclookup=mT{’lookup’}.find(’Did’:Did,’Vid’:Vid);
22. output=ACset(Acclookup, data, permission,’check’);
23. Return output
```

In the first line of algorithm 1, ACC will perform different operations according to the selected action. When you select Create, a new MT is created from lines 3 to 6, but does not contain any access permission settings, so we can install it on the blockchain. When we add new access permission settings to an ACC, lines 14 to 18, add specific permissions to the control layer's permission setting form Acclist.
When we want to delete a certain permission to an existing ACC, an existing permission setting will be deleted from the control layer's permission setting form Acclist from lines 16 to 19. In lines 21 to 22, the permission query results for data visitors will be returned.

In algorithm 1, we use acset function to set the specific permissions, and its logic can be described by algorithm 2.

**Algorithm 2. Access Control Configuration Function (ACset).**

Input: Acclist, data, permission, action  
Output: flag  
1. Switch action  
2. Case: add  
3. Row=lookup.find('data':data,'permission':permission);  
4. If Row>0  
5. flag=0  
6. lookup.append('data':data, 'permission': permission,'lasttime':time());  
7. flag=1:  
8. Case:delete  
9. Row=lookup.find('data':data,'permission':permission);  
10. If Row<0  
11. flag=0  
12. Lookup.remove('Row':Row);  
13. flag=1  
14. Case: check  
15. If Row<0 flag=0;  
16. Else flag=1  
17. Return flag

### 3. Privacy Access Control Model Based on Blockchain Smart Contract

Based on the ACC introduced in the previous chapter, we propose a new privacy access control model based on ACC. In this model, the access process of distributed storage system interacts with smart contract system to control privacy access. The model has two modules, access permission configuration module and access authority verification module.

In the access permission configuration module, data owners record their access permission configuration on the blockchain by creating new access control contracts. In the access authority verification module, data visitors provide their own identity information and access data through the interface provided by the smart contract. Base on the access permission configuration created by data owner and identity information provided by the data visitor, the smart contract determines whether the access is successful or not. This process can be described by algorithm 3.

**Algorithm 3. Data Visit Function (visit).**

Input: Vid, data, permission  
Output: message  
1. While ACC=SearchmT('data') do  
2. Did = findDid(ACC,data);  
3. flag = ACCfun('ACC':ACC,'Did':Did, 'Vid':Vid, 'permission':permission, 'action':check);  
4. If flag=1  
5. Message=Output(data); break;  
6. Message=Output("access denied");  
7. End while  
8. Return message
The SearchmT() function is used to traverse the MT in the blockchain and return the Did containing the same data, and output() is used to output the query results. Line 1 to line 2 of algorithm 3, the visit function will continuously traverse MT on the blockchain system and return the Did containing the given data. Then, ACC will be called in line 3 to query the access rights. If the query flag bit returns 1, indicating that the permission is allowed, the accessed data will be returned to the visitor through the output() function, otherwise, the inquirer will be prompted to "access is forbidden".

4. Experiment and Evaluation
Because the blockchain system is built on multiple network nodes, its performance can be easily affected by the number of nodes in the network. In this chapter, we will establish the blockchain system through fabric [10], and verify the performance of the proposed method through experiments. Existing works on access control based on blockchain technology focus on Internet of things [11,12], and they use Ethereum as the blockchain system. Since different data access rights have no significant impact on the efficiency of access rights verification, we will compare our method with these Ethereum-based method. We artificially create some access rights data, record them on the chain, and then test the efficiency of the verification.

In the experiment, we use fabric platform to build blockchain network. Fabric is an open source blockchain project initiated by the Linux foundation, which is committed to developing commercial blockchain platforms. Fabric contains the basic modules of blockchain, such as storage node, consensus mechanism and smart contract. It can allow different organizations to join the network and set up a dedicated channel to save their transaction records. At the same time, it also provides a special certificate authorization module. Fabric CA can be used to record and verify nodes and their identities. In this experiment, the network nodes in the blockchain are implemented through the ubuntu20.04 virtual machines, each virtual machine is allocated 2G memory and 1 core processor.

We have built three data storage systems, including 2, 6, 12 data owners respectively. Each data owner holds 5 items of data and sets access rights for 10 visitors. We use fabric to build a blockchain network with three organizations, each of which is used to store the access control table of a data storage system. We dynamically adjust the number of sorting nodes of each mechanism, and test its actual performance.

Our test divide into two parts: access control information writing part and access control verification part. In the access information writing part, each data owner writes a new access right to the specific data of a specific user to the blockchain network every 1 second, and tests the relationship between the writing time of all access control information and the number of sorting nodes in the network. In the access control verification part, one user verifies his access rights through the smart contract every 1 second. We record the time relationship between the time when 1000 users' access rights are verified and the number of sorting nodes in the network.

The experimental results of the access control information writing part can be represented by Figure 2, and the access control verification part can be represented by Figure 3. It can be seen from Figure 2 and Figure 3 that our method is more efficient than these Ethereum-based methods on both access control writing and verification scene. Moreover, when the number of sorting nodes is less than 40, the average time for writing and verifying access control information is less than 1.2 seconds, while the time interval between writing and verifying data is 1 second, so the actual response time is very small. Therefore, our scheme is efficient and feasible.
Figure 2. Experimental results of access control writing. The abscissa is the number of sorting nodes, and the ordinate is the average time (in seconds) of all access control writing.

Figure 3. Experimental results of access control verification. The abscissa is the number of sorting nodes, and the ordinate is the average time (in seconds) for all access validation.

5. Summary
In this paper, we propose a new access control model based on blockchain smart contract. It enables the data owners in the distributed storage system to record the data access rights on the blockchain through the smart contract, so as to prevent malicious tampering and single point of failure. When the data is accessed through the distributed storage system, the distributed storage system will interact with the blockchain system through the smart contract, query the access rights, and judge whether the access is allowed.

Acknowledgements
This work was supported by Science and Technology Project of State Grid Shanxi Electric Power Company, Project No 52051C19006S.
References
[1] Reed B, Chron E G, Burns R and Long D D 2000 IEEE. Mic 20 49-57
[2] Howard J H, Kazar M L, Menees S G, Nichols D A, Satyanarayanan M, Sidebotham R N and West M J 1988 ACM. Tcs 6 51-81
[3] Weil S A, Brandt S A, Miller E L, Long D D and Maltzahn C 2006 Ceph: A Scalable, High-Performance Distributed File System Proc. Symp. on Operating Systems Design and Implementation (Seattle, Washington) pp 307-20
[4] Cattaneo G, Catuogno L, Sorbo A D and Persiano P 2001 The Design and Implementation of a Transparent Cryptographic File System for UNIX Pro. 2001 USENIX Annual Technical Conference (Boston, MA) pp 199-212
[5] Nakamoto S 2008 Bitcoin: A peer-to-Peer Cash System Online Available
[6] Magyar G 2017 Blockchain: solving the privacy and research availability tradeoff for EHR data: a new disruptive technology in health data management Proc. IEEE 30th Neumann Colloquium (Budapest, Hungary) pp 135-40
[7] Zyskind G Nathan O and Pentland A 2015 Decentralizing Privacy: Using Blockchain to Protect Personal Data Proc. IEEE 36th Security and Privacy Workshops (San Jose, CA) pp 180-4
[8] Axon L and Goldsmith M 2017 PB-PKI: A Privacy-Aware Blockchain-Based PKI Proc. Int. Joint Conf. on e-Business and Telecommunications (Madrid, Spain) pp 311-8
[9] Yang L 2019 J. Industrial Information Integration 15 80-90
[10] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, and Yellick J 2018 Hyperledger fabric: A distributed operating system for permissioned blockchains Proc. 13rd EuroSys Conference (New York, NY) pp 1-15
[11] Ouaddah A, Elkalam A A, and Ouahman A A 2017 J. Scn 9 5943-64
[12] Novo O 2018 IEEE. Iotj 5 1184-95