A Distributed Storage System Based on Blockchain and Pipelined Code

Xuewei Li[1(✉)], Yue Chen[1], Yang Ba[1], Shuai Li[1], and Huaiyu Zhu[2]

[1] PLA Information Engineering University, Zhengzhou 450001, China
lxwyujin@qq.com
[2] 95937 Unit of PLA, Fuxin 123004, Liaoning, China

Abstract. To achieve data reliability and security in distributed storage systems, different coding schemes have been proposed. Encoded data would be stored in different nodes which are called hosts. However, the knowledge of hosts is learned by each other, and all of it must be known by the central server that stores meta-data, so that users and hosts cannot keep anonymity which reduces the difficulty of eavesdropping attack. The central server is a bottleneck due to single point of failure or low performance. These problems reduce the stability and the security of the storage system. To address these problems, we proposed a decentralized secure distributed storage system based on pipelined code and blockchain. In our scheme, we use blockchain to realize decentralization to solve the single point of failure problem and anonymity to data breach. An explicit pipelined code process combined with blockchain is presented to construct the storage function basis. Furthermore, a time-based replica storage mechanism is used in our scheme to achieve better trade-off between storage overhead and decode computation overhead.

Keywords: Security · Anonymity · Decentralized storage · Blockchain · Pipelined code

1 Introduction

In the big data era, distributed storage system (DSS) is used to meet large-scale data storage demand [1]. Commercial DSSs make a great role in people's daily life, such as Google file system (GFS) [2], Amazon S3 [25] or Hadoop file-system (HDFS) [26]. In these DSSs, replica mechanisms are used to strengthen reliability and availability of data and erasure code schemes are used to deduce storage overhead. Storage schemes that combine replica mechanism and erasure code schemes are usually introduced to achieve better trade-off between storage overhead and computation overhead [3, 4] in DSSs. Meanwhile, the security problems of DSS draw widespread concern due to the DSSs' nature. On the one hand, DSS runs in an open environment which leads its components to be attacked by adversaries easily. On the other hand, DSS data hosts storing the user data and the central server storing the meta-data are the main targets of adversaries. Furthermore, DSS data hosts need to learn the knowledge of each other in the process of encoding or decoding. That is, the data hosts cannot keep anonymity. The open environment and non-anonymity can be used by eavesdropping attack and

G. Wang et al. (Eds.): iSCI 2019, CCIS 1122, pp. 625–638, 2019.
https://doi.org/10.1007/978-981-15-1301-5_49

collusion attack and lead to data breach. In addition, the central server needs to undertake all route and computation work in the process of encoding and decoding. If the server is compromised by adversaries or has poor performance, it becomes a bottleneck of the DSS.

In the past 10 years, the security of DSS has been well studied by researchers. First, to solve the problem of eavesdropping attack, a lot of coding schemes [5–7] have been proposed, such as regenerating code schemes, cryptography-based code scheme [12]. Sensible coding schemes increase the difficulty of data breach through eavesdropping attack. Second, to solve the problem of single point of failure (SPoF), server replica scheme and decentralized system [12–14] have been widely studied. Server replica or decentralized technology migrate the central server work to different nodes to deduce system crash rate comes from SPoF. With the development of blockchain technology [15, 27], a new way which uses blockchain to realize decentralized DSS has attracted a lot of attention due to the blockchain's good performance in decentralization. It is easy to achieve anonymity and solve the SPoF problem with the anonymity and decentralization characteristics of blockchain in DSS.

The overarching goal of our study is to establish a secure DSS, in which users and data hosts can keep anonymity and the system does not suffer from the problem of SPoF. In this paper, we propose a decentralized secure distributed storage scheme, a pipelined code scheme based on blockchain, and a time-based replica storage mechanism. Specifically, we mainly focus on the following questions: (a) how to keep anonymity between users and each data host? (b) how to construct a coding scheme based on blockchain? (c) how to enhance the robustness of the system? The main contributions of this paper are three-fold.

- We introduce the framework of our decentralized DSS which is constructed by a two-layer blockchain and pipelined code.
- We provide a generic code construction that present the explicit process of pipelined code combined with blockchain.
- We show a time-based replica storage mechanism which achieves better trade-off between storage overhead and decode computation overhead.

The rest of the paper is organized as follows. We discuss related work in Sect. 2. In Sect. 3, we describe our main work from three aspects. And in Sect. 4, we discuss the security of our proposed scheme. Section 5 is the conclusion.

2 Related Work

When the personal data is outsourced in the DSS, a system running in an open environment, lots of security problems need to be concerned. The high value data and open environment make the data breach and SPoF be the main problems we need to address. Researchers have proposed many methods, such coding schemes, replica mechanism and blockchain-based DSS. The following Table 1 gives a summary that related work's performance in different aspects.

Table 1. Related work and their performance in different aspects.

	Sample	Anonymity	Anti-SPoF	Robust	Communication bandwidth	Computation overhead	Storage overhead
Coding schemes	[5–11, 14]	Weak	Weak	Strong	High	High	Low
Replica mechanism	[12, 14, 23]	Weak	Strong	Weak	Low	Low	High
Blockchain-based DSS	[15–19]	Strong	Strong	Strong	High	High	High

2.1 Coding Schemes

Data breach is a common problem in DSS due to its nature. To address it, lots of coding schemes have been proposed. Coding schemes used in DSS aim to provide a storage-efficient alternative to replication [14] in the beginning. In the same time, they play an important role when it comes to the data breach problem of DSS. Ankit et al. proposed a secure coding scheme and first investigated security in the presence of colluding eavesdroppers [5], and achieved the upper bounds on the secure file size and establish the secrecy capacity for any (l_1, l_2)-eavesdropper with $l_2 \leq 2$. In the same time, they employed cooperative regenerating codes to face multiple simultaneous node failures and eliminated the information leakage to the eavesdropper [6]. Shah et al. [7] provided constructions of regenerating codes that achieve information-theoretic secrecy capacity in the sitting where an eavesdropper may gain access to the data stored in a subset of the storage nodes and possibly to the data downloaded during repair of some nodes. To realize better trade-off between secrecy and system reliability, paper [8] provided explicit code constructions against eavesdropping attack and adversarial attacks in repair dynamics setting. Ravi et al. [9–11] investigated two types of wiretapping scenarios is DSS: (a) Type-I (node) adversary which can wiretap the data stored on any $l < k$ nodes; and (b) Type-II (repair data) adversary which can wiretap the contents of the repair data that is used to repair a set of l failed nodes over time. And they realized the optimal characterization of the secure storage-vs-exact-repair-bandwidth trade-off region of a (n, k, d)-DSS, with $n \leq 4$ and any $l < k$ in the presence of both Type-I and Type-II adversaries.

Coding scheme can solve the eavesdropping problem to some extent. But its shortcoming is obvious that it greatly increases computation overhead and communication bandwidth. And it is weak when it comes to collusion in DSS. It is expensive and weak to block data breach just through coding scheme in DSS.

2.2 Replica Mechanism

In [12], Lin et al. addressed the privacy issue of DSS with secure decentralized networked erasure code combined a threshold public key encryption scheme. This system is fully decentralized which introduces a new way to address another problem of DSS,

SPoF. SPoF may show up in the meta-data server when we encode or decode data. To some extent, the low performance of a server would lead to system crash. And the meta-data server is the important target of adversaries due to its high value, which may leak to adversaries more information when it is compromised. Cassandra [13] is a decentralized DSS which provides scalability, high performance, and wide applicability. Paper [14] proposed a decentralized erasure coding process that achieves the migration in a network-efficient manner in contrast to the traditional coding process.

Replica mechanism is a good method to address the SPoF problem. But it cannot achieve a good balance between SPoF problem and data breach. If decentralization cannot be implemented in the proper way, it increases the risk of data breach in the contrary.

2.3 Blockchain-Based DSS

With the development of blockchain technology [15], a new way to realize secure and decentralized DSS has drawn a lot of attention [18, 19]. Metadisk [16] is a new model in which a blockchain can serve as the backbone for a distributed application. It allows integration with existing open-source projects in a modular fashion to simplify the development of a decentralized storage network. In addition, Metadisk can use its own cryptocurrency as a means to pay for and exchange storage space and bandwidth in a peer to peer network, which may harness the powerful free-market force of self-interest to drive the networks growth and efficiency. Blockstack [17] is a decentralized DSS which combines blockchain and cloud storage. In this system, blockchain serves as a manager to guarantee system stability and security in the control plane with a two-layer blockchain (Blockchain Layer and Virtualchain Layer).

Considering to the advantages of blockchain-based DSS, we construct a decentralized secure DSS based on pipelined code and blockchain. With the anonymity characteristic of blockchain, we realize anonymity between users and hosts to against data breach. And with the decentralization characteristic of blockchain, we realize decentralization in our system to against the SPoF problem.

3 Proposed Secure DSS

In this section, we describe the proposed secure DSS. We present the system framework at first. Then we describe the explicit process of pipelined code scheme and replica placement mechanism based on the system framework.

3.1 System Framework

Our DSS framework is a three-layer architecture as shown in Fig. 1.

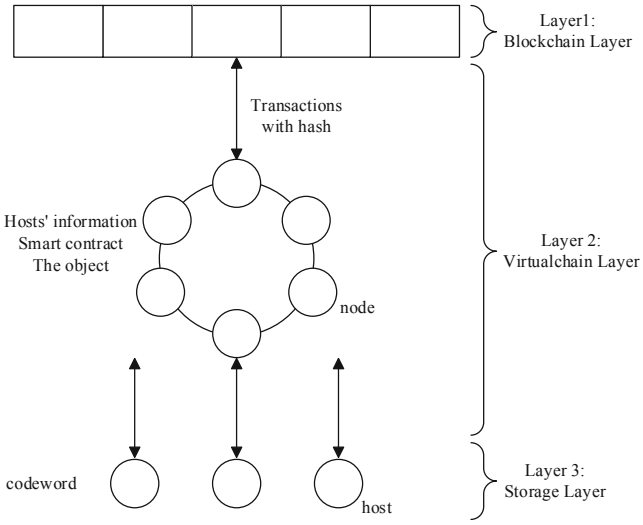

Fig. 1. Overview of our scheme architecture

The top two layers, Blockchain Layer and Virtualchain Layer, are in the control plane. The bottom layer, Storage Layer, is in the data plane. Users store or restore data in Storage Layer through Virtualchain Layer. Virtualchain Layer storages its own hash value in Blockchain Layer to protect its integrity. The characteristics of different layers are described as below.

Layer 1: Blockchain Layer. The Blockchain Layer is the security cornerstone of our system. It protects the security of Virtualchain Layer through storing the Virtualchain Layer's hash value as a transaction data. We take public chain, like Bitcoin's blockchain, as the underlying blockchain to maintain the Blockchain Layer. Public chain has lots of excellent characters, such as anonymity, tamper resistance and full historical record, that are useful for a secure DSS. In this paper, we use Bitcoin's blockchain as the underlying blockchain to construct our DSS. The reason is that Bitcoin's blockchain uses proof-of-work (PoW) as its consensus mechanism, which is known as the safest consensus. Another reason why we use Bitcoin's blockchain is that it is the safest public blockchain so far. It has been running for several years safely and steadily.

Layer 2: Virtualchain Layer. The Virtualchain Layer is the core functional module of our system. It realizes most main functions of our DSS, such as restoring hosts' information and smart contract, encoding and decoding user data and relaying on anonymous communication between users and hosts. It's a permissioned blockchain and uses fast consensus (e.g., Conflux [21]) to get high throughput. With fast consensus, even just a leader election algorithm [22], the Virtualchain can generate blocks to form ledger as fast as we need. So that it can record the information of hosts, metadata and user's operation history log in transactions as fast as we need. The SHA-1 hash of Virtualchain would be sent to Blockchain Layer periodically as a transaction. The Blockchain Layer stores the SHA-1 hash permanently in order that we can check

the integrity of the Virtualchain Layer. Smart contract stored on Virtualchain records the access control mechanism of user. When user wants to store/restore data, he sends a transaction to trigger the smart contract. Then the node who stores the smart contract carries out the code/decode process according to the access control mechanism. In addition, some nodes in Virtualchain Layer would be chosen as hosts to store the replica data temporarily according to the time-based replica storage mechanism.

Layer 3: Storage Layer. This layer is composed of data hosts, which store the true data of users. The hosts learn nothing of each other, and they just communicate with the Virtualchain Layer nodes chosen by bandwidth-aware [23]. They release their information (e.g., IP, storage capacity) to the Virtualchain Layer. After verifying these information, the Virtualchain Layer nodes record them. According to these information, the Virtualchain Layer nodes can choose suitable hosts to store data blocks in the encode process. The process of receiving and returning data blocks between users and hosts is also going through the Virtualchain Layer nodes. In this way, we keep the anonymity between users and hosts. For further protection of the security of hosts and the user data, we use one-time pad scheme to encrypt data, similar to [18]. After receiving the data blocks from the Virtualchain Layer nodes, hosts store them in its own storage system without any change.

3.2 Pipelined Code

Our pipelined code scheme is inspired by the RapidRAID [20], which is a pipelined erasure code used in DSS for fast data archival. We combine it with blockchain to realize an anonymous blockchain-based code scheme to construct the storage function basis. We depict an example of our code scheme using the code parameters (8, 4) in Fig. 2. An object $O = (o_1, o_2, o_3, o_4)$ is encoded into an 8-dimensional codeword $C = (c_1, \ldots, c_8) = (O, c_5, \ldots, c_8)$ using a classical systematic (8, 4) erasure code, that is, each o_i is a string of l bits. Every operation is performed using finite field arithmetic in the simulations of our scheme, and we use $\mathbb{F}_{2^8}$ as the finite field to deduce the computational complexity. But we do not insist on the code being systematic in the pipelined encode process. The host i (denoted by h_i on the figure) stores a replica of the raw data block $c_i = o_i$, $i = 1, \ldots, 4$. And hosts 5 to 8 store a second replica of the same object as well. The pipelined encoding process proceeds as follows. The host h_i sends o_i to node n_i, $i = 1, \ldots, 8$. Then the first node n_1 sends a multiple of o_1 to the second node n_2. The node 2 computes a linear combination of this multiple of o_1 with o_2, which h_2 sends to n_2, and forwards the result to n_3. The node 3 has now data o_3 from h_3, and again computes a linear combination of o_3 with which it received from n_2. The process is iteratively repeated from n_i to n_{i+1}, $i = 1, \ldots, 7$. And we get the final linear combination of all codewords from n_8 with which users can get the raw data.

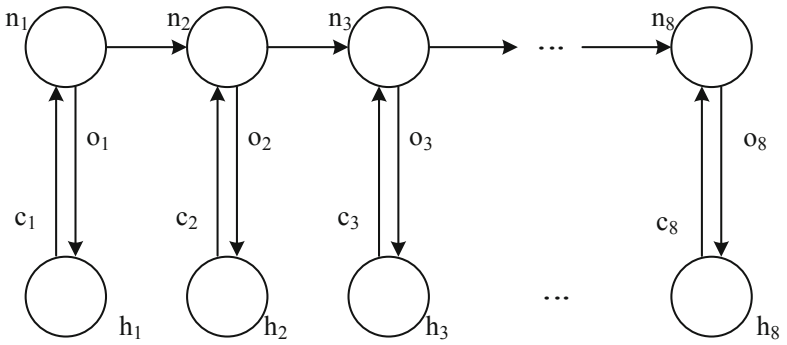

Fig. 2. An example of pipelined code process

At the same time, every node also generates a redundancy block c_i, based on what it receives, and forwards c_i back to h_i to update the codeword, $i = 1, \ldots, 8$. The generated blocks $C = (c_1, \ldots, c_8)$ constitute the final codeword. A formal description of this pipelined encode process is provided in Algorithm 1. Note that we use $x_{i,i+1}$ for the data which is forwarded from node i to node $i+1$, and $\psi_i, \xi_i \in \mathbb{F}_{2^l}, i = 1, \ldots, 8$ are predetermined values (Table 2).

Table 2. The pipelined code process with codeword generated by the (8, 4) erasure code.

Algorithm 1. Pipelined code process

1: **for** $i = 1, \ldots, 8$ **do** // hosts send codeword to nodes
2: (h_i sends o_i to n_i)
3: **end for**
4: $x_{1,2} \leftarrow o_1 \cdot \psi_1$ //nodes do pipelined code and forward it to successor
5: **for** $i = 2, \ldots, 7$ **do**
6: $x_{i,i+1} \leftarrow x_{i-1,i} + o_i \cdot \psi_i$
7: (n_i sends $x_{i,i+1}$ to n_{i+1})
8: **end for**
9: $c_1 \leftarrow o_1 \cdot \xi_1$ //nodes calculate codeword and return it back to hosts
10: **for** $i = 2, \ldots, 8$ **do**
11: $c_i \leftarrow x_{i-1,i} + o_i \cdot \xi_i$
12: (n_i sends c_i to h_i)
13: **end for**

The decode of object O can use linear coding notation $G \cdot O^T = C^T$, where G is composed by the predetermined values, ψ_i and ξ_i.

In our scheme, the pipelined code process requires little computation resources of every node. Because the process has fundamental difference with the normal datacenter operations and the system's computation overhead is shared by many nodes in the

Virtualchain Layer, that the system assigns different nodes to take the computation work evenly from different user in the pipelined code process. Bandwidth occupancy will only increase when user restore the data and the replica of the data stored on the node has been deleted. We show a typical application scenario in Fig. 3. The total network bandwidth increases to maximum value 8 blocks in the beginning of the code process. Then it decreases to 2 blocks and stays stable. Node 2, a representative node in the Virtualchain Layer, just undertakes small bandwidth of the maximum value 2 blocks in the whole process. Although the total network take a large network traffic, each node in Virtualchain Layer just loads a few in the pipelined code process. It is not possible that the problem of network congestion appears in our system.

3.3 Time-Based Replica Storage Mechanism

Erasure code scheme and replica mechanism are coexist to get better trade-off between storage overhead and computation overhead in most DSS. For the same purpose, we propose a time-based replica storage mechanism in company with pipelined code scheme in our DSS.

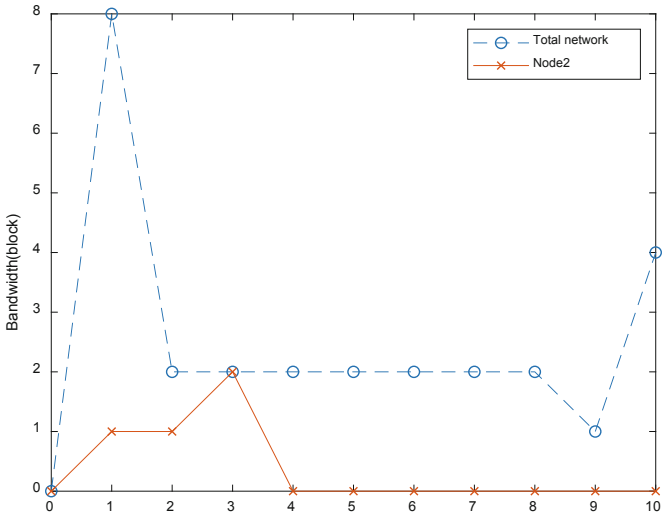

Fig. 3. Bandwidth Occupancy in the pipelined code process

We store the object' replica in a Virtualchain Layer node when the first time the user uploads the object or user reads the object. We take the object as hot data which the user may read it again within a short time. And for the hot data, we store its replica to reduce the computation overhead when user reads it the next time. A clock t and a threshold T are set and stored in the node along with the replica. The parameter value of the clock t is zero at beginning and it increases along with time. If user reads the object once again when $t < T$, he could get the object from the Virtualchain Layer node

directly and the parameter value of t will be reset to zero in the same time. The computation overhead is reduced to zero and the storage overhead is increased by four blocks. Through this way, we deduce the computation overhead at the cost of storage overhead. The replica stored in the node would be deleted when $t \geq T$, and the storage space is released. That is if user has not read the object for a time which is longer than T, we take the object as a cold data which the user would not read it in a long time. And for the cold data, we delete its replica to release the storage space.

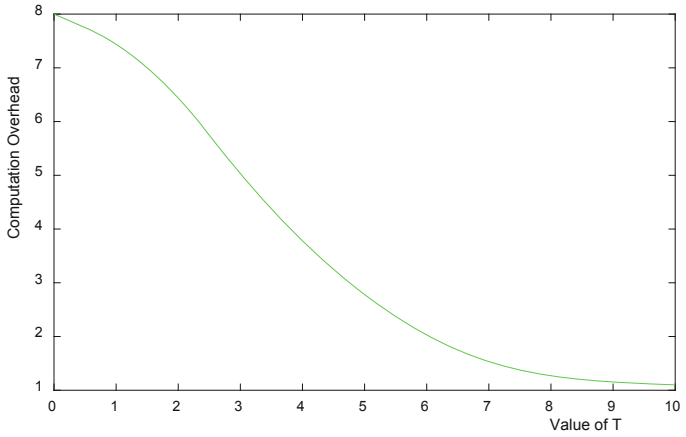

Fig. 4. The total computation overhead of the whole network changed with the value of T. When the value of T is big enough, the computation overhead decrease to very low.

The value of T is depended on the time interval t between user accesses the data tow times and the source value of the network. We need to keep a value C for the available computing resources of the network and a value S for the available storage resources of the node that storages the replica. We update the value of T when user accesses the replica:

$$T_{new} = \frac{t + T_{old}}{2} \cdot \frac{\alpha C_{new} + \beta S_{new}}{\alpha C_{old} + \beta S_{old}} \cdot \lambda \tag{1}$$

where $\alpha, \beta \in [0, 1], \alpha + \beta = 1, \lambda \in (0, +\infty)$, α and β denote the weights of computing resources and storage resources for network. If we want to reduce computation overhead, we can set $\lambda > 1$ to get a bigger value of T, that we can store the replica of the object for longer time. If user accesses the object during $T_{old} < t < T_{new}$, he can get it from its replica without computation. If we want to reduce storage overhead, we can set $0 < \lambda < 1$ to get a smaller value of T, that we store the replica of the object for shorter time. As shown in Fig. 4.

If the object has not been accessed until $t \geq T_{new}$, its replica stored in the node would be deleted, instead of $t \geq T_{old} \geq T_{new}$. In this way, we can get a better trade-off between storage overhead and computation overhead. As shown in Fig. 5.

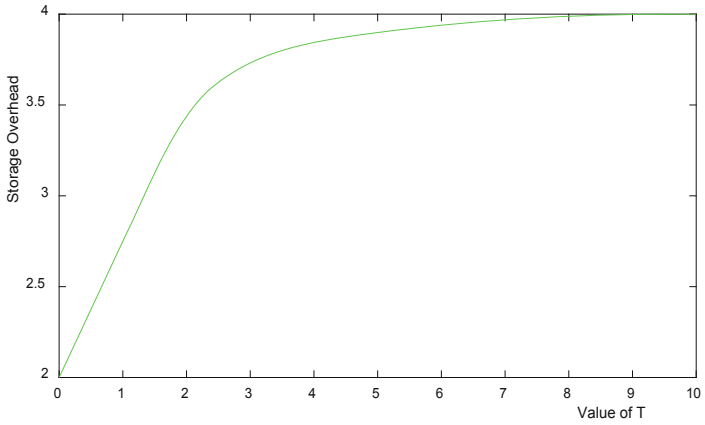

Fig. 5. The total storage overhead of the whole network changed with the value of T. When the value of T increases, the total storage overhead would increase.

In the same time, the time-based replica storage mechanism can further avoid network congestion. When a lot of users access their data in the same time, they can get the data directly from the replicas stored in the nodes. Nodes do not need to transmit blocks to calculating the data. In this way, the traffic of every node and the total network is reduced.

In a typical application scenario, such as the RapidRAID, its computation overhead and storage overhead remain at a fixed value respectively. Without the time-based replica storage mechanism, network congestion appears frequently in the RapidRAID when many users access their data in the same time. Compared with it, our system has a better performance to face the challenge that lots of users access their data in the same time.

4 Security Evaluation

In this section, we evaluate the security of our DSS. We focus on three aspects of the system security. The first is the data breach caused by adversaries attack, including eavesdropping attack and collusion attack. The second is the anonymity between users and hosts. The third is the SPoF problem.

4.1 Data Breach

Our system is running in an open environment that any computer can access the system and choose to be a user or a host if it wants. Adversaries can take advantage of this to pretend as trusted users or hosts. Then, they can implement eavesdropping attacks, one type of passive attack, to get user data in the pipelined code process. In our system,

however, it is very hard for adversaries to detect the target hosts who store the codeword. Because the storage hosts that store the codeword are selected randomly. And it is impossible to find target hosts through traversal search. In addition, the one-time pad scheme will make the replay attack meaningless to adversaries. Thus adversaries can not get user data even they can access our system as a user or host.

Collusion is one type of active attack which is prone to appear in DSS. There are three types of collusion in our system, hosts collusion, nodes collusion and collusion between nodes and hosts. First, as to hosts collusion, hosts in Storage Layer just communicate with nodes in Virtualchain Layer and they learn nothing of other hosts' knowledge. Therefore a host cannot communicate with another host and make collusion attack. Then, in our system, the message transformed between nodes and hosts and the message between different nodes will be formed a transaction and be written in the Virtualchain. Then the SHA-1 hash value of the Virtualchain will be sent to Blockchain Layer and recorded. The message will be permanently recorded as a tamper-resilience log due to the characteristic of blockchain. We can detect timely when nodes collusion and the collusion between nodes and hosts take place. And we can find out the malicious nodes or hosts at the same time. Thus collusion attack can not be implemented successfully.

From these above discussions, we can learn that adversary attack and collusion attack can be detected timely and blocked. It is very difficult for adversaries to steal the user data in our system.

4.2 Anonymity

In our system, the connection model between users and hosts is shown in Fig. 6. When the user wants to outsource his data to the system, the Virtualchain Layer plays a security-mediator role between users and hosts as the server does in [24]. The user encrypts the data with one-time pad key to keep the data's security before he sends it to the nodes and hosts. Then he sends the data O and smart contract SC, (O, SC), to the Virtualchain Layer nodes. On receiving the tuple, the Virtualchain Layer nodes encode O to $C = (c_1, \ldots, c_8)$ according to SC. Next, the system selects nodes from the Virtualchain Layer to store (O, t, T) and selects eight hosts from the Storage Layer to store C according to SC. When the user wants to restore the data form hosts, he needs to send a transaction to trigger the SC stored on the Virtualchain Layer. And then the node would decode C to O and return it to user.

We can know that users communicate with hosts anonymously and indirectly in the store and restore process, that the messages are relayed by the Virtualchain Layer nodes. So that the user anonymity is preserved. There is no message transmission between different hosts in the pipelined code process or decode process. And a host learns nothing of other hosts' knowledge. A host does not know other hosts' information even the existence of them. So that the host anonymity is preserved too.

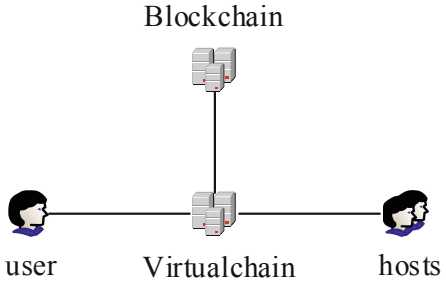

Fig. 6. The connection between user and hosts

4.3 SPoF

In our DSS, we take the Virtualchain Layer as a security-mediator to take the place of a central server. So the workload undertaken by the central server is shifted to the nodes in the Virtualchain Layer. When one user is outsourcing data through one node, another user can fulfil his own requirements through other nodes in the Virtualchain Layer. The computation overhead and communication overhead can be evenly shared by all Virtualchain Layer nodes. So that we do not need to worry about the poor performance of the central server.

Let us consider the situation that one node in the Virtualchain Layer is compromised by adversaries or permanently offline. In this situation, the codeword stored on it can be reconstructed through erasure code scheme and the replica stored on it can be stored on another node in the next time the user restores the object. Meanwhile, the bad node would be found and excluded from the Virtualchain Layer. The workload done by the bad node would be naturally shifted to other nodes. So that the problem comes form compromised central server does not exist in our DSS.

In a word, there is no SPoF in our DSS. The Virtualchain Layer can perform well in the place of a central server.

5 Conclusion

In this paper, we combine pipelined code and blockchain to form a decentralized secure distributed storage system. The proposed scheme employs double-chain structure, encryption and pipelined code in order to outsource the user data securely. In the proposed scheme, we present an explicit process of pipelined code scheme combined with blockchain, which could realize anonymity between hosts. And the blockchain is used to take the place of a central server, which can better solve the SPoF problem and realize the anonymity between user and host. In terms of performance, our system is robust and realize great trade-off between computation overhead and storage overhead.

In the future we will continue developing and evaluating other schemes that combine erasure codes and blockchain to form blockchain-based DSS.

Acknowledgements. This work is supported by the Key Technologies R & D Program of Henan Province (172102210017).

References

1. You, P., Huang, Z., Peng, Y., et al.: Towards a delivery scheme for speedup of data backup in distributed storage systems using erasure codes. J. Supercomput. **75**(1), 50–64 (2019)
2. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP) (2003)
3. Li, J., Li, B., Li, B.: Mist: efficient dissemination of erasure-coded data in data centers. IEEE Trans. Emerg. Top. Comput. **7**, 468–480 (2018)
4. Hou, H., Shum, K.W., Chen, M., et al.: BASIC codes: low-complexity regenerating codes for distributed storage systems. IEEE Trans. Inf. Theory **62**(6), 3053–3069 (2016)
5. Rawat, A.S., Koyluoglu, O.O., Silberstein, N., et al.: Optimal locally repairable and secure codes for distributed storage systems. IEEE Trans. Inf. Theory **60**(1), 212–236 (2013)
6. Koyluoglu, O.O., Rawat, A.S., Vishwanath, S.: Secure cooperative regenerating codes for distributed storage systems. IEEE Trans. Inf. Theory **60**(9), 5228–5244 (2014)
7. Shah, N.B., Rashmi, K.V., Kumar, P.V.: Information-theoretically secure regenerating codes for distributed storage. In: 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011, pp. 1–5. IEEE (2011)
8. Pawar, S., El Rouayheb, S., Ramchandran, K.: Securing dynamic distributed storage systems against eavesdropping and adversarial attacks. IEEE Trans. Inf. Theory **57**(10), 6734–6753 (2011)
9. Tandon, R., Amuru, S.D., Clancy, T.C., et al.: On secure distributed storage systems with exact repair. In: 2014 IEEE International Conference on Communications (ICC), pp. 3908–3912. IEEE (2014)
10. Tandon, R., Amuru, S.D., Clancy, T.C., et al.: Distributed storage systems with secure and exact repair—new results. In: 2014 Information Theory and Applications Workshop (ITA), pp. 1–6. IEEE (2014)
11. Tandon, R., Amuru, S.D., Clancy, T.C., et al.: Toward optimal secure distributed storage systems with exact repair. IEEE Trans. Inf. Theory **62**(6), 3477–3492 (2016)
12. Lin, H.Y., Tzeng, W.G.: A secure decentralized erasure code for distributed networked storage. IEEE Trans. Parallel Distrib. Syst. **21**(11), 1586–1594 (2010)
13. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. ACM SIGOPS Oper. Syst. Rev. **44**(2), 35–40 (2010)
14. Pamies-Juarez, L., Oggier, F., Datta, A.: Decentralized erasure coding for efficient data archival in distributed storage systems. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, Rudrapatna K., Sinha, P. (eds.) ICDCN 2013. LNCS, vol. 7730, pp. 42–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35668-1_4
15. Crosby, M., Pattanayak, P., Verma, S., et al.: Blockchain technology: beyond bitcoin. Appl. Innov. **2**(6–10), 71 (2016)
16. Wilkinson, S., Lowry, J., Boshevski, T.: Metadisk a blockchain-based decentralized file storage application. Technical report, hal, pp. 1–11, Storj Labs Inc. (2014)
17. Ali, M., Nelson, J., Shea, R., et al.: Blockstack: a global naming and storage system secured by blockchains. In: 2016 USENIX Annual Technical Conference (USENIX ATC 2016), pp. 181–194 (2016)

18. Fukumitsu, M., Hasegawa, S., Iwazaki, J., et al.: A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain. In: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), pp. 803–810. IEEE (2017)
19. Chen, Y., Li, H., Li, K., et al.: An improved P2P file system scheme based on IPFS and Blockchain. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 2652–2657. IEEE (2017)
20. Pamies-Juarez, L., Datta, A., Oggier, F.: RapidRAID: pipelined erasure codes for fast data archival in distributed storage systems. In: 2013 Proceedings IEEE INFOCOM, pp. 1294–1302. IEEE (2013)
21. Li, C., Li, P., Xu, W., et al.: Scaling Nakamoto consensus to thousands of transactions per second. arXiv preprint arXiv:1805.03870 (2018)
22. Aniello, L., Baldoni, R., Gaetani, E., et al.: A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database. In: 2017 13th European Dependable Computing Conference (EDCC), pp. 151–154. IEEE (2017)
23. Shen, J., Gu, J., Zhou, Y., et al.: Bandwidth-aware delayed repair in distributed storage systems. In: 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2016)
24. Chen, F., Xiang, T., Yang, Y., et al.: Secure cloud storage meets with secure network coding. IEEE Trans. Comput. **65**(6), 1936–1948 (2015)
25. Amazon.com, “Amazon S3”. http://aws.amazon.com/s3
26. Apache.org, “HDFS”. http://hadoop.apache.org/hdfs
27. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf