A Novel Approach for Improving Accuracy for Distributed Storage Networks

Liu Lu, Ke Yuanyuan, and Yuan Yong

Abstract With the development of storage technology and Internet technology, cloud storage continues to make its impact. Scalability, reliability, and lowered costs have made cloud storage widely used with success in businesses and individuals. The advent of the blockchain has brought some changes. As the incentive layer for IPFS, Filecoin allows storage resources to become tradable, greatly extending storage capacity. However, the process of testing the integrity of data still needs constant improvement. In this chapter, we propose a new data audit proof, in which nodes continuously upload hashed data that has been added to random numbers, and the smart contract will compare the result to verify the integrity of the data. Meanwhile, data owner could calculate and then challenge to verify the data integrity. There are audit miners responsible for regulating the behavior of miners and the protection of users' data, and audit miners in a state of semi-participation. It is demonstrated later in the chapter that this proof is accurate enough and resistant to attacks.

Keywords Distributed storage networks · Cloud storage · Blockchain

1 Introduction

Storage technology has evolved rapidly over the last few decades, with continuously decreasing hard disk prices and ever faster data speeds. However, the rapid growth of the online economy and big data technology has caused the need for data storage to expand exponentially, leading to the idea of cloud storage, in which data will be stored on cloud servers provided by third parties, and thus users can access data in a timely

L. Lu · K. Yuanyuan · Y. Yong (✉)
School of Mathematics, Renmin University of China, Beijing, China
e-mail: yong.yuan@ruc.edu.cn

L. Lu
e-mail: liulu0309@ruc.edu.cn

K. Yuanyuan
e-mail: ke_yy@163.com

Z. Zheng (ed.), *Proceedings of the Second International Forum on Financial Mathematics and Financial Technology*, Financial Mathematics and Fintech,
https://doi.org/10.1007/978-981-99-2366-3_4

and more convenient manner. Typically, cloud storage providers use technologies such as distributed storage (Mattson et al., 1970) to significantly reduce storage costs. However, the centralized storage makes cloud storage providers vulnerable to single points of failure and can create risks such as overstepping provider privileges and causing information leakage. Efficient centralized storage provisioning will remain mainstream in the future, but there is also an emerging and urgent need to meet users' needs for information security.

Traditional cloud storage providers, such as Amazon and Google, build cloud storage architectures with vast resources, using distributed storage technology to serve billions of users. Distributed storage means that data are spread across multiple storage servers and these scattered storage resources form a virtual storage device, effectively storing data in various places across the provider. The benefits of distributed storage are increased system reliability, availability, and access efficiency, as well as improved scalability. But the disadvantages are also obvious. We store our data on Google Cloud Drive on the basis that we trust Google to protect our data from being tampered with or lost, which can also lead to other disadvantages. The central server is vulnerable to attacks from adversaries, and internal failures and malpractice can also lead to data loss. As such, the security of cloud storage has also been a focus of attention in recent years. Traditional symmetric encryption algorithms put the keys on a central server, which makes it easier for attackers to get these keys and thus reduces the security of information. Moreover, data integrity verification whether data are stored efficiently and without deletion is also a crucial part of cloud storage services. Literature (Priyadharshini, 2012) summarizes the data integrity verification of traditional cloud storage, which is performed by TPA (Third Party Auditor) between the user and the CSP (Cloud Service Provider) to validate the data. The user poses a challenge to verify the integrity of their cloud data, and the TPA responds by comparing the original data, or the hash value of it according to literature (Zikratov et al., 2017), to verify the integrity. However, inefficiencies and tripartite or joint evil behavior can make opaque audit proofs unreliable. The convenience of centralized services brings with it the corresponding pitfalls. With the emergence of Bitcoin, decentralized technology continues to be improved, and decentralized storage brings an important addition to the traditional storage market.

The idea of providing decentralized storage has become popular with the rise of blockchain technology, and their combination could be considered a perfect fit. Blockchain enables reaching the consensus among decentralized, untrusted nodes. Its development has facilitated intensive research in several technologies such as cryptography, data structures, and consensus algorithms. When data are stored in multiple copies on the hard drives of different nodes, we cannot guarantee that all nodes are trustworthy. How to ensure the security and integrity of the data is a very crucial issue. After ensuring the stability of the storage, we also need to consider how to motivate people to become nodes and provide their own storage capacity, which requires a reasonable incentive mechanism.

Much of the current research is focused on issues such as access control, integrity verification, data retrieval, and traceability. Many platforms that offer distributed storage have already been launched. For example, the Sia

(Vorick & Champine, 2023) storage system, which was online earlier, has been unable to be developed effectively due to its less than optimal incentive design. IPFS (Benet, 2014), as a relatively complete platform, is a distributed storage system protocol for distributing and storing resources of various data types. Filecoin (Protocol Labs, 2023), as its incentive layer, incentivizes storage miners and retrieval miners to complete their own work by issuing tokens. Taking Filecoin as an example, there are three roles in Filecoin: client, storage miner, and retrieval miner. Clients pay for the service of storing and retrieving data. They can choose from a selection of available service providers. If they want to store private data, they need to encrypt it before submitting it to the service provider. Storage miners store clients' data for a reward. They decide for themselves how much space to provide for storage. After the client and the storage miner have reached an agreement, the miner is obliged to provide proof of their stored data on an ongoing basis. Everyone can view this proof and make sure that the storage miner is reliable. Retrieval miners give data to customers upon their request. They can retrieve data from clients or storage miners. Retrieval miners and clients use small payments to exchange data and tokens. The data are fragmented and the client pays a small amount of tokens for each fragment. Retrieval miners can also act as storage miners at the same time.

We will now show how a decentralized storage network stores and audits. As shown in Fig. 1, we demonstrate a cloud service with blockchain participation in two aspects: storage and audit. Data owners upload their data to miners on the server, who store the data and record the transactions on the blockchain. The blockchain also verifies data owner's information and protects the user's privacy. In order to ensure that their data are stored intact on the server, data owner challenges the TPA, which

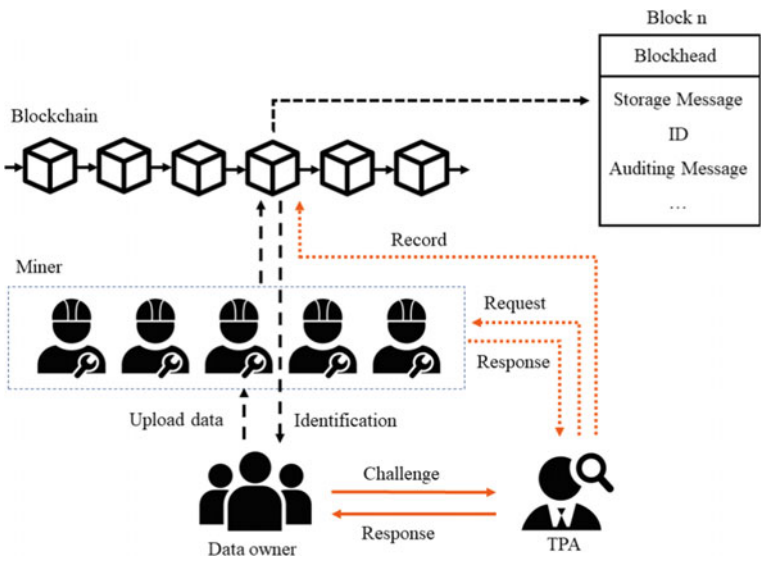

Fig. 1 Decentralized storage network framework

sends a request to the system and verifies the data provided by the miner in response to data owner's challenge. The verified result is then recorded on the blockchain. So each block in the blockchain stores information such as the height of the block, the block header, information about the previous block, a timestamp, storage message, ID, and Auditing message.

Data integrity verification in distributed storage, i.e., the red lines in Fig. 1, is our concern. Data integrity verification is the verification that data is stored intact in the storage space of each untrusted node. This affects the security of the data and is key to the availability of the storage service. Current data integrity validation can be divided into two kinds, one for traditional cloud-based data integrity validation and the other for blockchain-based data integrity validation. In turn, audit solutions using blockchain technology can be divided into whether or not TPA is involved. Most of these audit schemes verify against raw data and avoid dishonest behaviors such as delayed audits, sybil attack, and generation attack through consensus and incentive mechanisms. Blockchain-based data integrity verification can be used not only for auditing cloud data in cloud storage networks but also for different data scenarios to improve the security of the system. However, efficiency and accuracy cannot be achieved together in the process of decentralized data auditing. This will be described in Sect. 3. Most of the verification schemes that have worked better in current research do not run in public blockchains or require the participation of trusted central nodes. Instead, in fully decentralized blockchains, most are more efficient in order to ensure availability. However, the accuracy of verification cannot be fully guaranteed and the system is vulnerable to dishonest attacks. Our algorithm will improve long-term efficiency and stability in a fully decentralized blockchain with guaranteed accuracy.

Our work is based on a modification of Filecoin for verifying the integrity of data in distributed storage. The audit miner in this algorithm is semi-involved and determines whether the data are kept intact by comparing the hash values of the data shards. If the result does not meet the desired goal, the audit miner will first ensure the integrity of the data and then find the storage miner that created the problem, acting as a reasonable supervisor. In Sect. 2, we will summarize the past work on distributed audit algorithms and describe the characteristics of each platform. In Sect. 3, we will present our audit algorithm and analyze its advantages and the problems it solves. Sect. 4 will analyze the fault tolerance of this audit proof.

2 Related Works

2.1 Audit Research

Ensuring data integrity in cloud computing has always been an important issue, and it is a guarantee that cloud computing can be widely used. Traditional data integrity verification can be divided into deterministic and probabilistic types. The dishonest behavior of TPA is also an important issue for audit algorithms when they are entrusted to perform audit integrity verification work. Blockchain technology

with a decentralized architecture no longer relies excessively on the honesty of third parties and reaching an overall consensus based on a reasonable consensus and incentive mechanism and then a mutual benefit for all parties is the core element to be explored at this stage.

Literature (Zikratov et al., 2017) proposes a private blockchain called Zeppar, which determines the integrity of data by comparing the hash values of files. The use of cryptographic techniques to verify data integrity by comparing the original data is a common and applicable method. Such an approach is also used in literature (Wei et al., 2020), where smart contracts monitor data changes based on the unique hash value corresponding to the file generated by the Merkle Hash Tree (MHT). Verifying data integrity by constructing MHT is a relatively convenient method, e.g., in literature (Bai et al., 2018; Li et al., 2020). In literature (Li et al., 2020), data owner (DO) stores the verification tag of the data on the blockchain and verifies the data integrity by constructing MHT. After the blockchain network receives a request from the DO, it calculates the MHT root of the specified data, the CSP receives the DO's challenge and also calculates the corresponding MHT root, and the DO verifies the integrity of the data by comparing the two. We can find that neither the method of comparing file hash values nor the construction of MHT requires the involvement of TPA. Such an approach can be very efficient for verification but will compromise on the degree of centralization or be less fault-tolerant. It is relatively suitable for distributed storage systems where efficiency is required.

In order to ensure the activity of the data, some auditing schemes use the provision of random numbers to avoid users falsifying the results of data validation in advance. Literature (Pinheiro et al., 2020) uses the user's data information to generate random challenges and uses the smart contract to audit the challenge-response information sent by the CSP. The audit scheme also assesses the trustworthiness of each CSP.

2.2 Distributed Storage Project

- Sia: A relatively early decentralized storage platform, Sia in literature Vorick and Champine (2023) enables storage contracts to be formed between peer-to-peer nodes. The contracts are stored in the blockchain, making them publicly auditable. Sia divides files into 30 parts, encrypts each part using the Threefish algorithm, and distributes them to different nodes. Reed-Solomon erasure coding makes it possible to fully recover a file by requiring only any 10 of the 30 parts. With Merkle Tree (Ralph, 1988), nodes are required to upload storage proofs (Maxwell, 2023) within a certain time frame or be penalized.
- Filecoin: Literature (Benet, 2014) proposes a distributed peer-to-peer web protocol: IPFS (InterPlanetary File System). Based on a content addressing protocol, it makes network transmission faster, content storage easier and nodes protection safer. Filecoin can be considered the incentive layer of the IPFS system, providing decentralized cloud storage in the form of tokens distributed in a rational way. Its audit algorithm Proof-of-Replication shown in literature (Protocol Labs, 2017)

deferred encoding of data to get a copy of the data and then generates a zero-knowledge proof to guarantee the correctness of the encoding process. Its other consensus algorithm, Proof-of-Spacetime, requires miners to periodically generate Merkle proofs for the replicas and submit them to the blockchain compressed with zero-knowledge proofs for tokens reward. Such an incentive encourages miners to store data correctly and to prove data liveness to obtain proof of work as a reward.

- Arewave: Arweave cloud storage platform is similar to Filecoin in that it features a service that provides permanent storage. It has designed a new consensus algorithm, Proof of Access, which is based on the concept that new blocks require random validation of previous blocks. This turns the original blockchain into a network of blocks, where nodes no longer need to store exponentially growing amounts of data, but only certain data, allowing the data to be distributed evenly across the system to achieve distributed storage.
- Storj: Storj Labs (2018) built at Kademlia is not a fully decentralized cloud storage system and it is dedicated to data storage durability and storage quality. Satellite nodes act as fully trusted nodes in storj for data management and data integrity review. The data are sliced after encryption and the data integrity is guaranteed by Proof of Retrievability (Juels et al., 2007) consensus algorithm. The satellite nodes are responsible for communication between the user and the storage node, for storing metadata for the user, as well as auditing and enforcing Proof of Retrievability. The presence of the satellite nodes makes storj resistant to Byzantine attacks, but at the expense of the network's performance, resulting in poor scalability.

Table 1 has given the difference among these four platforms. We can find that their audit proofs are different and lead to other differences in other natures.

However, there are still some flaws. The current work almost verifies the integrity of distributed storage data under specific conditions, but none of it has a systematic analysis of the limitations of auditing. We will analyze the compromise factors that

Table 1 Distributed storage networks comparison

	Degree of decentralization	Storage location	Consensus algorithm	Audit proof
Sia	Fully	Off chain	Proof of work	Proof of storage (Maxwell, 2023)
Filecoin	Fully	Off chain	Expected consensus (Protocol Labs, 2017)	Proof of replication, proof of spacetime
Arweave	Fully	On chain	Proof of work, proof of access	Proof of access
Storj	Satellite nodes exist	Off chain	Proof of work, proof of stake	Proof of retrievability (Juels et al., 2007)

can arise from audit algorithms in distributed storage in the next section. We also analyze what requirements the Filecoin platform should have for auditing and what constraints it should have on storage miners. We design an audit proof for distributed storage and prove that it is sufficiently accurate and fault-tolerant.

3 Audit Algorithm

3.1 An Audit Framework

In this chapter, we will reformulate the audit proof of Filecoin to address the current problems of Filecoin platform. Our goal is to retain the decentralized nature and allow the distributed storage network to complete the audit process on its own. Audit miners will only appear when necessary. This will ensure the accuracy of the audit and improve the efficiency of all nodes in reaching consensus on the audit results. We propose the audit impossibility proposition regarding the distributed storage networks as follows:

Proposition 1 (Audit impossibility): The degree of decentralization, the accuracy of audit results, and audit efficiency cannot be reached at the same time.

When integrity checks are performed on an absolutely centralized storage server, CSP can invest significant resources in a way that increases the efficiency and accuracy of the audit, as many cloud storage providers do nowadays. This is the approach that currently dominates the cloud storage market. However, with decentralization, we cannot perform fast and efficient integrity checks on untrustworthy storage nodes based on today's computing power and the sheer volume of data. How to balance accuracy and efficiency is currently the key issue for auditing in all distributed storage. For Filecoin, decentralization is its biggest advantage. However, too frequent data auditing not only affects the accuracy of the data audit results, but also causes the system to be less stable when the nodes are offline. Therefore, to improve the efficiency of auditing while ensuring the accuracy of the audit results is the issue considered in this chapter.

Our design starts by slicing and numbering the data owners encrypted data using the shard technique and then generates multiple copies (k copies) by replication, which will be stored randomly on storage miners. When auditing these files, we will take the last 16 bits of the hash of the previous block as the new random number $\mathcal{N}$, which all miners will add to each of their stored shards for hashing. The result will need to be uploaded to the hash pool in a certain order with the miners' signatures. All the hash values are automatically matched by the smart contract. By determining whether the corresponding hash value is equal to k, it is concluded that the data are stored intact in the distributed storage network. This allows a simple comparison of

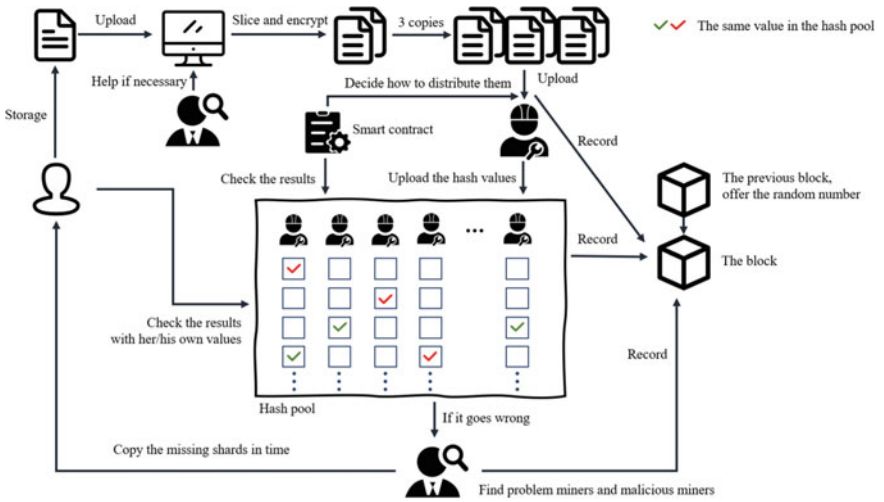

Fig. 2 Algorithm overview

the results to determine whether the data owners' data are completely stored across the network. The data owner can also, but not necessarily, add his/her own shard data to the random number $\mathcal{N}$ and hash them. The result is then compared across the hash pool to determine if the data are stored correctly on the miners by finding the same k values in the network result. If the storage miner is not validly stored, the audit miner needs to find the problem miner quickly and back up the data in time (Fig. 2).

There are three roles in our platform, data owners U, storage miners M, and audit miners A. Data owners upload encrypted data according to their needs and can challenge the integrity of the data. The storage miner stores the data sequentially as assigned by the smart contract as well as uploads proof of data integrity every once in a while. The audit miner is responsible for handling the distribution of data, as well as reviewing and supervising miners, protecting data integrity, regulating content, and assuming legal responsibility. The number of audit miners is limited and storage miners can be audit miners at the same time. Audit miners only appear if there are problems with the audit.

3.2 Data Uploading

From the moment the user uploads data, the user U_i should divide his/her data $D(i)$ into several shards by using slicing and encryption technology in order to keep the data secure. If he/she does not have enough computing power to handle too large data, he/she can upload them to audit miner A for slicing and encrypting and then pay some tokens. All the shards are then distributed by the audit miner to storage

miner M and back to user U_i. Data slicing is a common technique used in distributed storage to protect the data. We use $D(i, j)$ to denote the jth shard of U_i's data. The number of shards U_i has, J_i, will be determined by the size of U_i's data. Replication is also used to replicate k copies of $D(i, j)$: $D(i, j, k)$. Before uploading, the U_i can add a random number N known only to him/her to calculate the result for all $D(i, j)$ and encrypt the result as a validation audit option later. Next, the miner M_a is randomly sent a request to store the corresponding shard or not, with a specific request (Definition 1). M_a that receives the request has to choose whether to store the data or not, depending on its storage capacity. The miner who confirms storage will store the corresponding $D(i, j)$ on his local hard disk. The label (i, j) of the data $D(i, j)$ will only be stored in the smart contract and will not be transmitted to the miner who stored it. The miner will not know the exact label (i, j) of the data he/she stores, but will only number them sequentially according to the order in which he/she stores $D(i, j)$. If $D(i, j)$ is the sixth data storage of M_a, then the corresponding $D(i, j)$ is $M_a(6)$. We would use $M_a()$ to express the set of shards stored by M_a. This allows for better protection of the user's information and data, and prevents the exchange of content between miners as much as possible.

We can effectively prevent malicious miners from sybil attacks or other attacks by slicing and replicating the data and storing them in a decentralized manner. We also require an appropriate specific request for sending shards to avoid joint attacks by miners.

Definition 1 *(Request of distributing shards):* The distribution of the set $\{D(i, j, k), \forall i, j\}$ to miners is subject to the following principles:

1. The number of M_a storing the data of $D(i)$ cannot be less than half of J_i.
2. No miner M_a will receive two or more storage requests for a single copy of data $D(i, j)$.
3. No miner M_a will receive storage requests for $D(i, j)$ and $D(i, j + 1)$.
4. Miners M_a and $M_{a'}$ will not receive storage requests for $D(i, j)$ and $D(i, j^{'})$ together.
5. No miner will store more than y copies of $D(i)$.
6. Miners M_a and $M_{a'}$ will store no more than z identical shards in the shards pool.

This ensures that the data are stored in a sufficiently decentralized manner, with enough miners storing the data owner's data together, so that a single point of failure does not have a major impact on the overall storage. It also ensures that the user's data are not stolen in its entirety, guaranteeing the security of the data. Definition 1 also makes the data stored by the two nodes different, avoiding outsourcing attack. We will specifically analyze the effectiveness of our algorithm in Sect. 4.

3.3 Self-integrity Verification

Now all data owner's data has been uploaded to each storage miner. We then need to continuously interact with all miners to ensure that data liveness is guaranteed and

the data are being stored intact. This is the core of the work in this chapter. We have described in Sect. 2, the current auditing methods, both fully decentralized and not fully decentralized, that are able to do the job but not well in the accuracy of audit results or audit efficiency. This chapter proposes a solution that does not require the data owner's data to be compared and achieves self-auditing through self-comparison in the blockchain network, which substantially improves the long-term stability of the system. At the same time, our proof is more efficient and can quickly reach a consensus on the integrity of all data in a short period of time with responses from all nodes. We also allow data owners to initiate challenges and quickly check the integrity of their own data through the hash algorithm.

The blockchain network audits whether the storage miners have correctly stored the corresponding data within a period T. To ensure timeliness, we use the last 16 bits of the hash value of the previous block as a random number $\mathcal{N}$. After getting $\mathcal{N}$, the miner has to upload the result of the hash operation of all his shards and $\mathcal{N}$ together with his signature M_a_sign within a specified time T. Now we obtain a new set: $\{hash(M_a(), \mathcal{N}), M_a\_sign\}$ to express the result of the hash of all M_a's shards and its signature. It is important to note that the set is ordered, again according to the order in which M_a stores the shards. The advantage of this design is that even when faced with a pile of results, the smart contract can determine the corresponding label (i, j) based on its position. We will use $H(a, b)$ to denote the hash value corresponding to the bth shard of the miner a with $\mathcal{N}$, $D(i, j)\_hash$ to denote the hash value of $D(i, j)$ with $\mathcal{N}$ (Table 2).

After the storage miner M_a has uploaded his/her $\{hash(M_a(), \mathcal{N}), M_a\_sign\}$, the smart contract will quickly determine if the number of $H(a, b)$ is equal to the number of shards already stored by M_a, and if it does not match, invalidate this result and demand M_a to recalculate and upload the new result. If the result matches, the result is accepted and moves on to the hash pool. Next, the smart contract will compare the number of occurrences of all the results in the hash pool. If there are exactly k identical results, i.e., if there are k sets (a, b) s.t. all the results of $H(a, b)$ are equal, then it will be decided that all the copies of the shard have been stored correctly. This would be the best result that can be achieved. All the storage miners need to store their data correctly for their own benefits. If all miners store correctly, all the nodes can quickly and accurately obtain the result that the data are stored intact. We will now determine whether the data are stored correctly based on the occurrences of each hash value.

Definition 2 *(strong integrity)*: The number of occurrences of $D(i, j)\_hash$ is exactly equal to k.

Definition 3 *(weak integrity)*: The number of occurrences of $D(i, j)\_hash$ is greater than or equal to 2 and less than k.

Table 2 Notations for operations/implications

Symbol	Notations for operations/implications
U	Data owners
M	Storage miners
A	Audit miners
U_i	The jth data owner
$D(i)$	Data owner i's data
$D(i, j)$	The jth shard of U_i's data
k	The number of copies
J_i	The number of U_i's shards
$D(i, j, k)$	All of the U_i's shards
M_a	The ath storage miner
(i, j)	The label of $D(i, j)$
$M_a(6)$	The sixth shard stored by M_a
$M_a()$	The set of M_a's storage
T	Cycle time for storage miners uploads
N	The random number set by the user
$\mathcal{N}$	The random number from the previous block
M_a_sign	M_a's digital signature
$H(a, b)$	$M_a(b)$'s hash value with $\mathcal{N}$
$D(i, j)\_hash$	$D(i, j)$'s hash value with $\mathcal{N}$

If all shards achieve strong integrity, we can assume that the storage network has stored all data correctly and that all nodes would agree on this. If all shards achieve weak integrity, we can assume that all data are stored securely on the storage network. Weak integrity is a lower requirement for data availability in storage networks. During auditing, it is more of a constraint on the miners, so strong integrity is what is required by distributed storage networks.

We will now discuss what to do if strong integrity is not achieved. If the number of occurrences of a hash value is greater than k, the possible scenario is that the miners are jointly misbehaving with each other and copying the same result for output. This is because when the storage miner receives the shard corresponding to that result, no other shards are received, and only if the miner has stored other miners' shards. In this case, the $k + \alpha$ results are assigned a number (i, j) based on their location, and the numbers are then compared to find the miner with the incorrect result by audit miners A. The first step is to find the set of $\{(i', j')\}$ corresponding to the wrong hash value, and then check whether the number of occurrences of hash value is k. If it is k, the shard has been completely stored in the storage network. Otherwise, this

number can only be less than k, if so, A needs to find which miners did not upload the right value and ask them to upload in time. If they upload the wrong results, ask them to re-store correctly to solve the problem.

In fact, it is more often the case that the number is less than k. In case when the hash values whose number is less than k, we need the corresponding miners to upload proofs of the correct storage of the corresponding $D(i, j)$. The following results may occur:

1. The miner correctly stores $D(i, j)$ and uploads the correct hash result.
2. The miner correctly stores $D(i, j)$ but uploads the wrong hash result.
3. The miner incorrectly stored $D(i, j)$ but uploaded the correct result.
4. The miner incorrectly stored $D(i, j)$ and uploaded the wrong result.

Audit miners A need to immediately copy $D(i, j)$ to ensure that they couldn't be lost. After that, A will handle errant storage miners as above. Such handling effectively avoids errors caused by miners offline. We will also judge storage miners who make frequent errors as malicious miners. If for the same $D(i, j)$, all the results of the hash operation are different or it is not possible to distinguish the correctness of the result, then A can ask all miners storing the $D(i, j)$ to recalculate it with the random number N and compare it with the result calculated by U_i. In time, copy the data of the miners that output the correct result and ask U_i to re-add another random number N to the calculation and keep the result for future use (Fig. 3).

The above is the process by which a blockchain storage network audits of its own. This process allows for quick consensus to be reached under the condition that all the data are stored correctly, as well as finding malicious nodes if consensus is not reached.

3.4 Data Owner's Integrity Verification

After the data owner gets $\mathcal{N}$, he/she can also get a set of hash values $H(i, j)$ generated by U_i by performing a hash operation on his/her own data shards $D(i, j)$. Smart contract will look for k occurrences of these values in the hash pool to determine whether his/her data have been stored completely. If exactly each result occurs k times, then it is almost certain that U_i's data has been stored correctly. If not, then the storage miner in problem can be found quickly and the data copied by the audit miner in his/her storage in time. Such an audit approach improves the shortcomings of self-integrity verification and increases the accuracy of data integrity verification.

3.5 The Game of Miners Versus Storage Networks

Storage miners can only earn if they store the user's data correctly and upload $H(a, b)$ correctly. If the miner wants to earn without storing correctly, he needs to join with

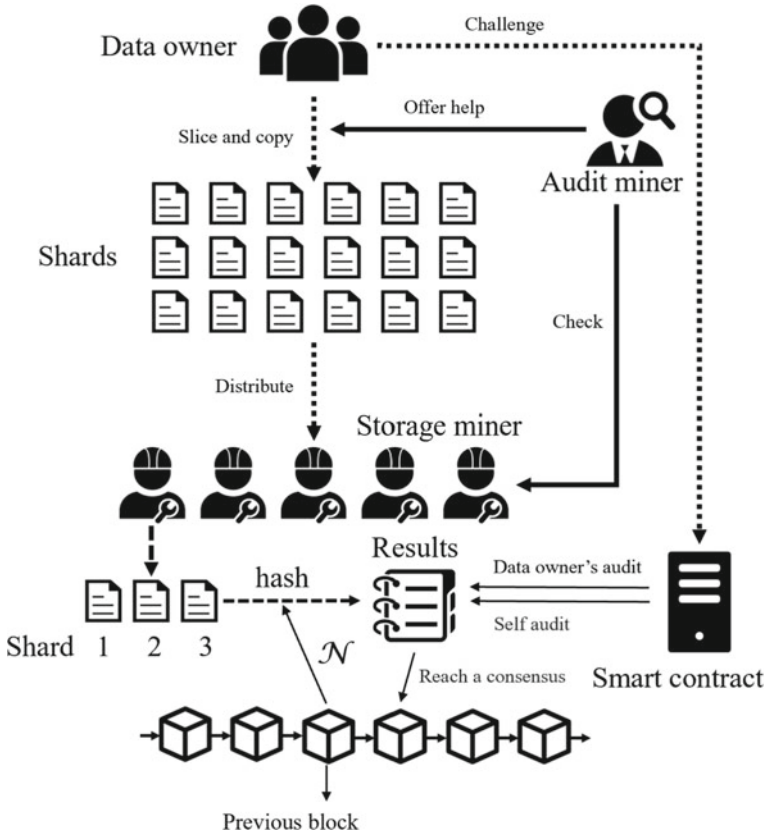

Fig. 3 Audit algorithm

other miners. The miner does not know the number (i, j) of the data he is storing, so he needs to send a request to all miners in the network. And other miners can be rewarded by reporting those malicious nodes. The union of storage miners does not earn a reward, only the individual fulfillment of the storage function makes the storage network maximize its benefits. For audit miners, audit miners are only given the appropriate audit access if there is a problem with the storage miner. Audit miners are only able to earn more rewards by continuously completing audit tasks and tasks delegated by data owners. These ensure that all miners are driven by profit to achieve stability.

Thus our system satisfies incentive-compatible property and also data integrity, recoverability, publicly verifiable, and auditability. The satisfaction of the five properties is obvious. These are the same properties that filecoin satisfies. We can say that our audit proof is reasonable.

4 Fault-Tolerance Verification

We now describe three attacks that are common in distributed storage networks.

- Sybil Attack (Douceur, 2002): Sybil attack is a type of attack in peer-to-peer networks in which a node in the network operates multiple identities actively at the same time and undermines the authority/power in reputation systems. In a distributed storage network, a malicious miner can create multiple sybil identities pretending to store many copies in order to be rewarded, but only one copy is stored in his local.
  In our proof, a miner cannot claim to have stored multiple shards, as the number of shards per share is limited to k. Meanwhile, there is a little additional gain for a malicious miner to pretend to store multiple copies by creating multiple identities. Since each miner stores different content and for two storage miners, they have the number of the same shards less than z. We control the revenue in such a way that storage miners will not receive enough benefit in creating a witch identity, making them less likely to take risks for it. Subsequently, we can limit such a situation even further by monitoring IP address, generating $M_a()$ proofs, etc. Such a scenario makes sybil attacks much less profitable.
- Outsourcing Attack: By relying on fast access to data from other storage providers, malicious miners promise to store more data than they can actually store.
  If a malicious miner wants to launch an outsourcing attack, the miner cannot know the shard number and can only determine if there is an overlapping shard by sharing the miner's $H(a, b)$ set with each other; if there is an overlapping shard, the hash result can be quickly retrieved later in the audit. But the benefit to the provider is weak, and the inclusion of an exposing mechanism keeps miners from going to extremes for the weak benefit. So we can conclude that the benefits of a small number of miners cooperating are much less than the risks associated with incomplete storage.
- Generation Attack: Malicious miners claim to have more storage than they actually have through a small program to gain a greater advantage in the mining competition.
  With slicing and cryptography, miners cannot effectively generate data with small program. The generated proof results need to be computed by hash function, and a small change can lead to a huge difference in results. There are strict penalties for generation attack in Filecoin, so this attack can be substantially avoided from an incentive point of view.

5 Concluding Remarks

In this chapter, we focus on current research on auditing and point out the imperfections of current auditing. We also analyze the audit requirements for Filecoin and redesign an audit algorithm for it. The algorithm determines whether the data have been stored intact in the storage network by comparing the results in the hash pool by means of storage miners uploading the hash results. The audit miner is set to a semi-participating state and will only join in time to gain access if a problem arises. Such an auditing algorithm is relatively accurate and secure for decentralized storage networks. Besides, it is obtained that the algorithm is highly fault-tolerant.

Our algorithm is not yet well designed in terms of incentives and needs to prove that the algorithm can be put into widespread use. Incentives are a key part of getting the algorithm used, and it is important to play the game between miners and the storage network so that both sides can get the optimal solution for their interests. The regulation of the data is also something that needs to be considered in the next phase. Our algorithm needs to be more complete in the future.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (72171230), and the Science and Technology Development Fund, Macau SAR (File No. 0050/2020/A1).

References

Bai, L. H., Xue, J. T., Xu, C. X., et al. (2018). DStore: A distributed cloud storage system based on smart contracts and blockchain. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer.

Benet, J. (2014). IPFS—Content addressed, versioned. *P2P File System*.

Douceur, J. R. (2002). The sybil attack. Springer.

Juels, A., Kaliski, B. S., PORs, J. (2007). Proofs of retrievability for large files. In *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS* (Vol. 07, pp. 584–597). ACM.

Li, J., Wu, J., Jiang, G., et al. (2020). Blockchain-based public auditing for big data in cloud storage. *Information Processing & Management, 57*(6), 102382.

Mattson, R. L., Gecsei, et al. (1970). Evaluation techniques for storage hierarchies. *IBM Systems Journal*.

Maxwell, G. Proof of storage to make distributed resource consumption costly. https://bitcointalk.org/index.php?topic=310323

Pinheiro, A., Canedo, E. D., Sousa, R., et al. (2020). Monitoring file integrity using blockchain and smart contracts. *IEEE Access, 8,* 198548–198579.

Priyadharshini, B. (2012). Data integrity in cloud storage. IEEE.

Protocol Labs. Filecoin: A decentralized storage network. https://filecoin.io/filecoin.pdf

Protocol Labs. Technical report: Expected consensus.

Protocol Labs. Technical report: Proof-of-replication.

Ralph, C. (1988). Merkle: A digital signature based on a conventional encryption function. In C. Pomerance (Ed.), *Advances in cryptology, CRYPTO* (Vol. 87, pp. 369–378). Springer.
Storj Labs. Inc. Storj: A decentralized cloud storage network framework.
Vorick, D., & Champine, L. Sia: Simple decentralized storage. https://blockchainlab.com/pdf/whitepaper3.pdf
Wei, P. C., Wang, D., Zhao, Y., et al. (2020). Blockchain data-based cloud data integrity protection mechanism. *Future Generation Computer Systems, 102,* 902–911.
Zikratov, I., Kuzmin, A., Akimenko, V., et al. (2017). Ensuring data integrity using blockchain technology. In *2017 20th Conference of Open Innovations Association (FRUCT)*.