Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6998046, 13 pages
https://doi.org/10.1155/2022/6998046

Research Article

Blockchain-Based Self-Auditing Scheme with Batch Verification for Decentralized Storage

Zhonghao Yuan, Jiaojiao Wu, Jianpeng Gong, Yao Liu, Guohua Tian, and Jianfeng Wang

School of Cyber Engineering, Xidian University, Xi'an 710126, China

Correspondence should be addressed to Jianfeng Wang; jfwang@xidian.edu.cn

Received 14 January 2022; Revised 22 March 2022; Accepted 24 May 2022; Published 26 June 2022

Academic Editor: Jinguang Han

Data owners outsource their data to remote storage providers without keeping local replicas to save their precious storage resources. However, the ownership and management of data are separated after outsourcing. How to ensure the integrity and recoverability of outsourced data becomes a significant problem. Provable Data Possession (PDP) and Proofs of Retrievability (POR) are two cryptographic protocols that enable users to verify the integrity of outsourced data. Nevertheless, the state-of-the-art PDP and POR schemes either need users to perform the complicated audit tasks by themselves or delegate these tasks to a Third-Party Auditor (TPA). Moreover, these schemes are constructed on a centralized storage framework which vulnerably suffers single-point-of-failure. In this paper, we propose a blockchain-based decentralized self-auditing scheme with batch verification. Firstly, data owners outsource their data to decentralized storage nodes, which can achieve self-auditing based on blockchain without TPA. Secondly, our scheme uses Pedersen-based polynomial commitment to significantly reduce the number of authenticators. Furthermore, we propose a batch verification algorithm, which can verify multiple proofs from different storage nodes to improve the verification efficiency. Finally, we analyze the security of our scheme and implement a gas-efficient system prototype using the smart contracts of the Ethereum Reposten test network. The results demonstrate that the scheme is practical.

1. Introduction

In recent years, cloud storage has become an essential application in our daily life. It is greatly convenient and flexible for users to store, update, and retrieve their data in the cloud [1]. Thus, more and more users outsource their data to cloud storage providers to save their local storage resources. However, the ownership and management of outsourced data are separated after users upload their data to the remote cloud. Therefore, there have arisen numerous security problems in the cloud storage. In 2015, a European data center of Google was affected by lighting strikes and permanently lost 100 GB data. In the same year, Tencent Cloud lost their customers' data, which caused significant losses for this company. Therefore, the cloud storage providers are not fully trusted. It is important for users to ensure the outsourced data are correct and intact.

Traditional cloud auditing schemes are mainly classified into two categories: Provable Data Possession (PDP) [2] and Proofs of Retrievability (POR) [3], which are two cryptographic protocols allowing users to verify the integrity of data without retrieving the data, and the POR protocol can also guarantee data recoverability. Concretely, the verifier can randomly sample a challenging set of original data, and the prover generates the corresponding integrity proof, which can be audited to ensure the integrity of the out-

sourced data. To avoid online and computational burden on data owners, existing schemes introduce a Third-Party Auditor (TPA) to help data owners to audit data [4]. However, TPA is a centralized entity that easily suffers from single-point-of-failure. In addition, many cloud auditing schemes assume that TPA will never collude with storage providers. This strong and impractical assumption may be easily broken driven by certain interests. Moreover, in the state-of-the-art auditing schemes, the data owner needs to generate homomorphic linear authenticators which grow linearly with the number of file blocks. These schemes cause an amount of computational overhead for data owners and storage redundancy for storage providers. In practical applications, we also desire to perform batch verification of multiple data auditing tasks to improve audit efficiency. Therefore, it is of great practical significance to design an efficient integrity auditing scheme without TPA.

With the rapid development of public blockchains, such as Ethereum [5]and Solana [6], more and more researchers focus on constructing a decentralized storage system by deploying blockchain. In these systems, data owners can outsource their sensitive data to decentralized storage nodes, which can arbitrarily join the network to contribute their idle storage resources. The existing blockchain-based auditing schemes [7–9] for decentralized storage can ensure the integrity of data without TPA and are naturally resistant to single point of failure due to the decentralized feature. However, these schemes delegate complex auditing tasks, including a lot of cryptographic operations, to the smart contract deployed in the blockchain, which cannot be efficiently implemented in Ethereum Virtual Machine (EVM) and may cause amounts of gas consumption. Thus, it is a challenging problem to handle the computation-intensive operations on smart contracts in blockchain-based auditing schemes for decentralized storage.

1.1. Our Contributions. In this paper, we propose a self-auditing scheme with batch verification based on blockchain for decentralized storage. Our scheme is based on blockchain technology, PDP protocol, Pedersen-based polynomial commitment, and batch opening polynomial commitment, to achieve efficient data self-auditing without TPA. In summary, our contributions can be listed as follows:

(i) We propose a blockchain-based self-auditing scheme with batch verification. We remove TPA and allow data owners to store their data to decentralized storage nodes, who can interact with the blockchain to achieve self-auditing

(ii) We adopt Pedersen-based polynomial commitment to construct the homomorphic linear authenticators, which significantly decreases storage overhead and slightly reduces authenticator computation time. In addition, we also propose a batch verification algorithm to verify multiple proofs simultaneously, which can improve verification efficiency

(iii) We implement a gas-efficient self-auditing system in the platform of the Ethereum test network and perform security analysis and performance evaluation. The results show that our scheme is efficient and practical

1.2. Related Work

1.2.1. Centralized Outsourced Storage. In 2007, Ateniese et al. [2] proposed the PDP protocol, which is the first public audit scheme to verify the data integrity in an untrusted server. Based on the homomorphic linear authenticators constructed by the RSA signature, data owners can probabilistically verify the integrity of data without retrieving original data. However, this scheme cannot guarantee data recoverability. In 2007, Juels and Kaliski [10] presented the POR protocol, which can achieve data recoverability by using the erasure coding, but their scheme only supports fixed number audits and cannot achieve public verifiability. In 2008, Shacham and Waters [3] improved POR scheme with provable security. Based on BLS signatures [11] and the bilinear pairing, the verifier can publicly verify the data integrity and recover the remote data at any time. Nevertheless, the above schemes cannot guarantee data privacy because the data is stored by plaintext in storage providers. At the same time, the data owners must always be online to audit the storage providers.

Then, there are numerous new protocols improving the PDP/POR system model, such as additional properties of privacy, multiple-replica, and batch auditing. Curtmola et al. [12] firstly proposed multiple-replica provable data possession (MR-PDP) to guarantee the data recoverability. MR-PDP scheme allows data owner that stores t replicas of a file to the cloud system to verify the multiple-file integrity. Then, there are numerous scheme attention on multiple-replica auditing [13–18]. Considering the data security, Wang et al. [4] proposed a privacy-preserving public auditing scheme that outsources the auditing tasks to TPA and supports batch auditing. Following, several schemes [19–21] concerned with privacy for data owners are proposed. To improve storage redundancy and communication overhead, Yuan and Yu [22] proposed a constant communication cost auditing scheme using the polynomial commitment for cloud storage. Besides, there are an increasing number of blockchain-based auditing schemes. Using the RSA signature, Wang et al. [23] proposed a blockchain-based private auditing scheme with small authenticators' redundancy. Their scheme can divide data into arbitrary blocks for generating authenticators which should be uploaded to the blockchain. Moreover, there are many schemes [7, 8, 24–26] that stored the verification proofs to blockchain to achieve undeniable verification interactions. Furthermore, Yuan et al. [7] and Wang et al. [8] used the smart contract to replace TPA to audit the storage providers. Recently, Su et al. [27] proposed a self-auditing scheme for multiple cloud servers without TPA. Their scheme stored the data to several cloud servers and achieved data integrity via the interactions of these servers. However, each server requires multiround interactions to acquire the other servers' proofs. Moreover, data owners need online to challenge servers.

1.2.2. Decentralized Outsourced Storage. All of the above schemes concern on centralized outsource storage framework, which has many obvious drawbacks. Firstly, centralized storage providers are vulnerable to single-point-of-failure making the users' data at risk. Secondly, it is too expensive to centralized storage compared with decentralized storage. In order to deal with the above problems, decentralized storage is becoming a hot spot.

Li et al. [28] proposed the notion of IntegrityChain, which is a decentralized storage system supporting MR-PDP. IntegrityChain is a blockchain that mainly stores the information of storage node registration, storage transactions, and auditing proofs. Based on the schemes [4, 22], Du et al. [9] also proposed a blockchain-based auditing scheme with privacy-assured in the decentralized storage network. However, due to the problem of Solidity language, currently, the smart contract cannot effectively support complex cryptographic primitives. At the same time, Francati et al. [29] utilized the blockchain nodes to store users' data. Miner audits the integrity of data when generating a new block. To ensure data recoverability, Chen et al. [30], respectively, distributed the data and its replica to cloud and decentralized storage providers. The cloud audits the replica stored in the decentralized storage nodes as TPA. With the development of blockchain technology, there are numerous blockchain-based decentralized storage projects. Both Swarm [31] and Storj [32] are decentralized storage networks that outsource audit services to centralized auditors. Unlike Swarm, based on an incentive system through the smart contract on Ethereum, Storj provides an incentive layer with cryptocurrency. Therefore, storage nodes may collude with auditors to deceive data owners. Sia [33] is a fully decentralized storage platform for uploading and downloading data between users and storage nodes. To verify the integrity of data, the storage node transfers Merkle proofs to blockchain and receives Siacoins as a reward. Unlike Sia, Filecoin [34] employs proof-of-spacetime and proof-of replication to guarantee that miners have correctly stored the committed data, which provides more robust storage security. However, the proof generation time and the computational overhead make it hard to be deployed.

1.3. Organization. This paper is organized as follows. Section 2 provides some notations and the cryptographic primitives used in our scheme. Section 3 introduces the system model and security goals of our scheme. We propose our main scheme in Section 4 and provide the security analysis in Section 5. Then, we evaluate our scheme performance and show the results in Section 6. Finally, we conclude this paper.

2. Preliminaries

In this section, we describe notations used in our scheme as Table 1. Then, some cryptographic primitives are introduced to construct our scheme.

2.1. Bilinear Map. There are three multiplicative cyclic groups, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, where the order of $\mathbb{G}_1$ is p. Let g_1 and g_2 be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. We use $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ to denote a bilinear map with the following properties.

Table 1: Notations and descriptions.

Notations	Description
p	Big prime
$\mathbb{Z}_p$	Cyclic group module safe prime p
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	Three different multiplicative cyclic groups
g_1, h_1	The generators of $\mathbb{G}_1$
g_2	The generators of $\mathbb{G}_2$
m	The number of file blocks
d	The number of file chunks
id	The file identifier
H_1, H_2	The two hash functions
t	The number of challenged set
k	The number of storage nodes
$P_i(x)$	The challenged polynomial
$Q_i(x)$	The quotient of challenged polynomial

(i) Bilinear: for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g^a, h^b) = e(g, h)^{ab}$

(ii) Computable: there exists a computable algorithm to compute the map e efficiently

(iii) Nondegenerate: $e(g_1, g_2) \neq 1$

2.2. Pedersen-Based Constant Size Polynomial Commitment. In a polynomial commitment scheme, the committer can commit a polynomial to a group element, and then, the committed polynomial can be opened at any point by a verifier. Based on an algebraic property of polynomial $f(x) \in Z[x]$: $(x - r)$ perfectly divides the polynomial $f(x) - f(r)$, $r \in \mathbb{Z}_p$, Kate et al. [35] proposed Pedersen-based polynomial commitment scheme which commits two polynomials simultaneously with the constant communication overhead. In their scheme, the proof generated by the committer proves that $\phi(r)$ is the evaluation of the committed polynomial at point r.

Firstly, we introduce the Pedersen commitment [36] that a value u with a random number r computes as:

$$C = \mathrm{com}(u, r) = g^u h^r, \tag{1}$$

where g and h are two generators of a cyclic group $\mathbb{G}$, whose order is p.

Concretely, Kate's scheme is described as below:

(i) Setup $(1^\lambda, s)$: given the security parameters λ and the degree of the polynomial s, a trusted entity generates private key $SK = \alpha$, selected randomly from $\mathbb{Z}_p$, and public key $PK = (\mathbb{G}, \mathbb{G}_1, g, g^\alpha, \cdots, g^{\alpha^{s-1}}, h, h^\alpha, \cdots, h^{\alpha^{s-1}})$. $\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative cyclic

groups, and the order of group $\mathbb{G}$ is prime p. g and h are two generators of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_1$ is a symmetric bilinear pairing

(ii) Commit $(PK, \phi(x))$: given the PK and a polynomial $\phi(x)$, the committer chooses a polynomial $\widehat{\phi}(x)$ of degree s from $\mathbb{Z}_p[x]$. the commitment is computed as $C = g^{\phi(\alpha)} h^{\psi(\alpha)} \in \mathbb{G}$

(iii) CreateWitness $(PK, \phi(x), \widehat{\phi}(x), r)$: given the $r \longleftarrow^R \mathbb{Z}_p$, the committer calculates $\psi(x) = (\phi(x) - \phi(r))/(x - r)$ and $\widehat{\psi}(x) = (\widehat{\phi}(x) - \widehat{\phi}(r))/(x - r)$. Then, the witness ω is calculated as $g^{\psi(\alpha)} h^{\widehat{\psi}(\alpha)}$ based on PK. Finally, the algorithm outputs $\{r, \phi(r), \widehat{\phi}(r), \omega\}$

(iv) VerifyEval $(PK, C, r, \phi(r), \widehat{\phi}(r), \omega)$: given the output of the algorithm CreateWitness, it is verifiable that $\phi(r)$ is the evaluation at the point r of polynomial which is committed by C as below:

$$e(C, g) = e\left(w_i, \frac{g^{\alpha}}{g^{i}}\right) e\left(g^{\phi(i)} h^{\widehat{\phi}(i)}, g\right). \tag{2}$$

2.3. Batch Opening Polynomial Commitment. To improve the verification efficiency of Kate's scheme [35], Boneh et al. [37] proposed two polynomial commitment schemes which can open proof for multiple points and polynomials at the same time. The first scheme is introduced to construct our scheme. The opening proof of their first scheme is constant size as same as Kate's scheme, but the verifier will have a large amount of computation if there are many distinct evaluation points. The concrete scheme is described as follows.

(i) Setup $(1^{\lambda}, s, t)$: s is the degree of the polynomial, and t is the max number of opening points. Then, a trusted authority uniformly chooses α as private key from $\mathbb{Z}_p$ denoted by SK and computes public key PK as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_1^{\alpha}, \cdots, g_1^{\alpha^{s-1}}, g_2, g_2^{\alpha}, \cdots, g_2^{\alpha^t})$. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicative cyclic groups where the order of $\mathbb{G}_1$ is p. g_1 and g_2 are the generators, respectively, selected from $\mathbb{G}_1$ and $\mathbb{G}_2$. e is an asymmetric bilinear pairing: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$

(ii) Commit $(PK, \phi(x))$: the algorithm outputs $C = g_1^{\phi(\alpha)}$

(iii) CreateWitness $(PK, r, \{\phi_i(x)\}_{i=1}^n, T = \{r_i\}_{i=1}^t, \{S_i \subset T\}_{i=1}^n)$: given a random $\gamma \in \mathbb{Z}_p$ sent by the verifier, several polynomials $\{\phi_i(x)\}_{i=1}^n$, and their individual opening point subset $\{S_i \subset T\}_{i=1}^n$, a prover computes the polynomial $h(x)$ as $\sum_{i=1}^n \gamma^{i-1} \cdot (\phi_i(x) - \phi_i(r_i))/Z_{S_i(x)}$, where $Z_{s_i(x)}$ is the polynomial of $\prod_{r_i \in S_i} (x - r_i)$. The witness ω is the polynomial commitment of $h(x)$, computed as $g_1^{h(\alpha)}$

(iv) VerifyEval $(PK, C, r, \phi(r), \widehat{\phi}(r), \omega)$: verifier computes $F = \prod_{i=1}^n e(\gamma^{i-1} \cdot (\mathrm{cm}_i - [r_i(x)]_1), Z_i)$ and verifies the following equation $F =^{?} e(\omega, g_2^{Z_T(\alpha)})$, where Z_i is a polynomial commitment of $g_2^{Z_{T \setminus S_i}(x)}$

3. Problem Statement

3.1. System Model. We propose a blockchain-based public self-auditing scheme that ensures data integrity and recoverability for decentralized storage. The framework of our scheme is shown in Figure 1, which obtains three roles: data owner, storage node that belongs to a decentralized network, and blockchain.

(i) Data owner (DO): DO outsources his data to several distributed storage nodes to save storage space. Before uploading data to nodes, DO processes data in advance to guarantee data privacy and recoverability

(ii) Storage node (SN): SN, a peer of the decentralized network, wants to outsource his storage resources to gain more interest. Our scheme assumes that several nodes that store the same DO data cannot collude with each other. SN should generate proof for each auditing task and interact with the blockchain

(iii) Blockchain (BC): due to the transparency and tamper-proof property, blockchain servers as a trusted third party in our scheme. In the auditing stage, BC will challenge SN and store the auditing proof generated by the SN

As shown in Figure 2, our scheme is outlined in detail. In the setup stage, the DO divides the outsourcing file F into m blocks, and each $2n$ blocks form a chunk. For each chunk, DO generates an authenticator. Then, in the storage stage, DO will distribute file chunks and corresponding authenticators to several SNs. During the self-auditing stage, each SN firstly calculates the challenged set via the information of the blockchain header. Secondly, each SN will generate the proof according to the challenged data chunks and transfer the proof to the smart contract. Finally, each SN gets the other node's proofs from the smart contract, verifies the correctness of all proofs, and transfers the audit report to the smart contract.

3.2. Threat Model and Security Goals. Considering the fairness of the scheme, we describe malicious behaviors as follows. Besides, we assume that at least one of the nodes that stored DO's data is honest. Firstly, the SN may delete data rarely accessed by DO to save storage costs for more interest or directly leave the decentralized network. Secondly, due to various accidents, such as hardware and software failure, the outsourced data stored on the SNs may be tampered or deleted. In order to his reputation, the SN may hide the facts of data loss until the time of data retrieval. Finally, DO may generate erroneous metadata to SNs for gaining more

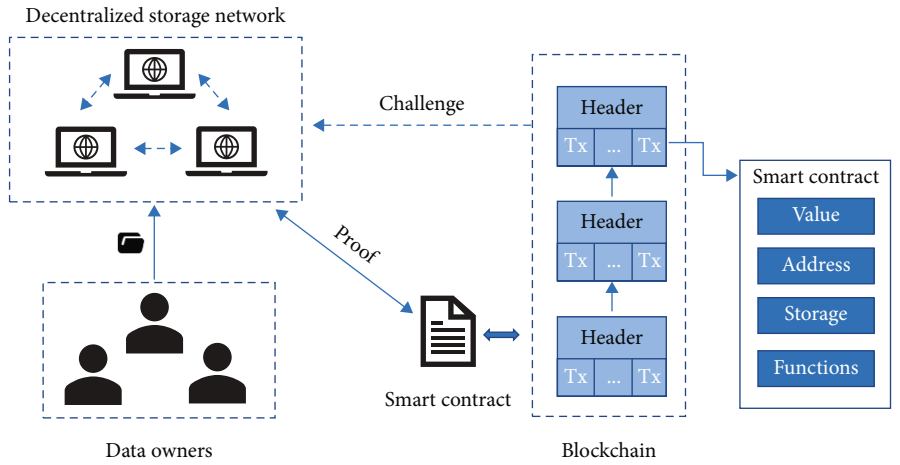

Figure 1: System model.

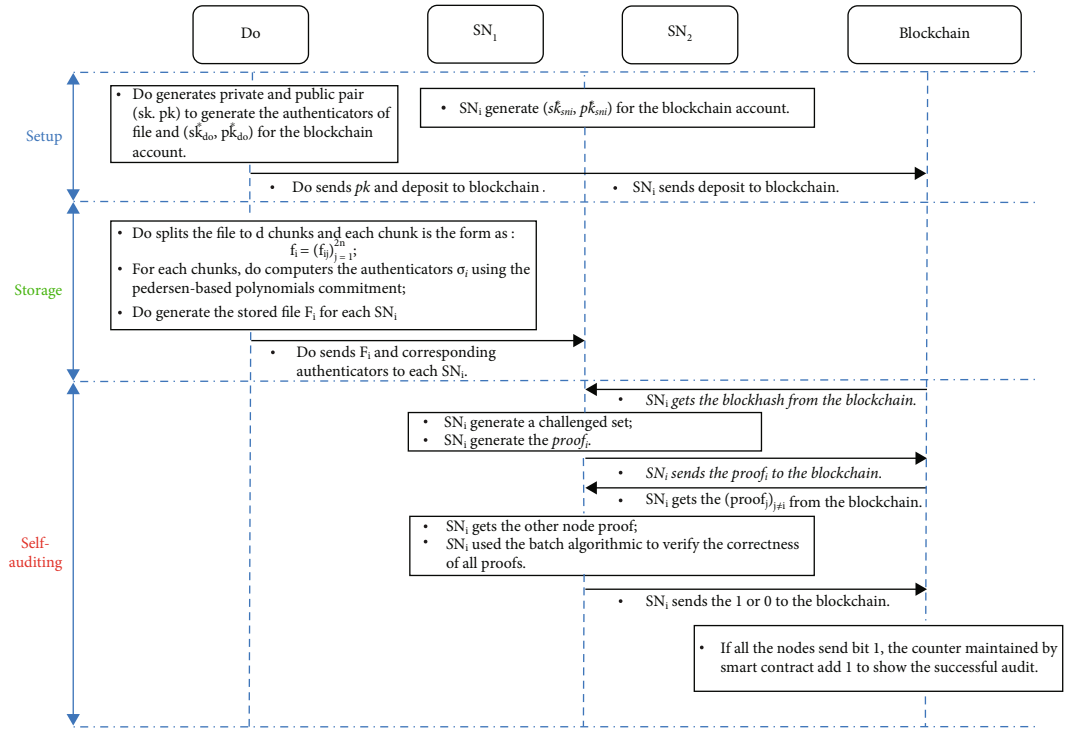

Figure 2: Overview of our proposed self-auditing scheme.

interests. In our scheme, we want to achieve the security goals as follows.

(i) Data privacy: we should protect DO's data privacy that SN and malicious adversaries cannot extract data contents

(ii) Storage correctness: we should guarantee that if SNs can pass each audit, they must correctly store the DO's data

(iii) Batch auditing: we require SNs to correctly verify multiple audit tasks at one time to improve the scheme's efficiency. If the batch auditing can be passed, SNs must correctly store the DO's data

(iv) Fairness: we should ensure the fairness of incentive mechanism, which will reward honest participants and punish malicious participants

4. Our Main Scheme

In this section, we first describe the formal definition of our scheme. Then, we propose the main idea of our scheme. In the end, we present our scheme in detail. The main algo-

rithms of our scheme are defined below. During the following presentation, we describe our scheme from the view of DO and SN_i.

(i) Setup $(1^\lambda, n, k)$: given the security parameters λ, the number of blocks in each chunk n and the number of opening points k. Then, this algorithm outputs private and public key pairs (SK, PK) for DO to preprocess the uploading file, and the blockchain accounts $\{sk^*_{sni}, pk^*_{sni}\}$ and $\{sk^*_{do}, pk^*_{do}\}$ for SN_i and DO

(ii) TagGen (SK, F): this algorithm inputs the DO's SK and uploading file F. It outputs the processed file $\widehat{F}$ and corresponding authenticators $\{\sigma_i\}_{i=1}^d$

(iii) FileDistribution(pk^*_{sni}, k): given the pk^*_{sni} and the number of storage nodes k, this algorithm outputs the file chunks and authenticators of each SN_i required for storage

(iv) ChallGen $(pk^*_{sni}, \text{blockhash}, t)$: given the pk^*_{sni}, the number of challenged set t and the blockhash on the blockchain, this algorithm outputs the challenged set of each SN_i, denoted by $C_i = \{\{c_j, d_j\}_{j=1}^t, r_i\}$

(v) Self-auditing $(PK, \{C_i\}_{i=1}^t, \widehat{F}, \{\sigma_i\}_{i=1}^d)$: given the PK, the number of challenged set and metadata, this algorithm outputs 1 or 0, where 1 represented this round audit is successful and 0 failure

4.1. Main Idea. The overview of our scheme is shown in Figure 2. We propose a blockchain-based self-auditing scheme to solve the problem that the traditional schemes introduced TPA. The main idea is that the storage nodes act as verifiers and interact with the smart contract to complete each audit. Concretely, the data owner first encrypts and encodes the file for data privacy and retrievability. Then, selecting several storage nodes, the data owner distributes part of the file to each node. For each audit, every node generates the proof through the information of blockchain and transfers the proof to the blockchain. Each node serves as a verifier to verify the proofs of other nodes. If every proof is passed the verification, each node transfers the successful message to the smart contract, such as 1. The smart contract automatically maintains a counter recording the number of successful audits. When the data owner retrieves the file, the smart contract sends the data owner's deposit to the storage nodes according to the counter.

4.2. The Concrete Scheme. We use three different multiplicative cyclic groups in our protocol, $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$. And the order of the group $\mathbb{G}_1$ is prime p. Let three generators g_1, h_1, and g_2, respectively, selected from $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ be a bilinear pairing. $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \longrightarrow \{0,1\}^\lambda$ are expressed as two collision-resistant hash functions. Besides, the stored file is denoted to F separated into m blocks. Each $2n$ blocks form a set of data for generating authenticators and saving storage space. It should be noted that each block is the element of group $\mathbb{Z}_p$ for the security of the audit. Furthermore, our scheme can support batch auditing where the number of audit proofs is k.

Setup$(1^\lambda, n, k)$. The data owner randomly selects two elements, α and υ from group $\mathbb{Z}_p$, as his private key and computes $\{g_1^{\alpha^j}, h_1^{\alpha^j}\}_{i=1}^{n-2}$, and $\{g_2^{\alpha^j}, g_2^{\upsilon(\alpha)^j}\}_{j=1}^t$ as part of the public key. Let $\varepsilon = g_2^\alpha$ and $\beta = g_2^{\alpha\upsilon}$ in order to conveniently express in a single audit. Therefore, the private key SK is (α, υ), and the public key PK is $(\varepsilon, \beta, H_1, H_2, g_1, g_2, h_1, \{g_1^{\alpha^j}, h_1^{\alpha^j}\}_{i=1}^{n-2}$ and $\{g_2^{\alpha^j}, g_2^{\upsilon(\alpha)^j}\}_{j=1}^t)$. Meanwhile, data owner and each storage node SN_i generate the private and public keys of the blockchain account, respectively, denoted by $\{pk^*_{sni}, sk^*_{sni}\}$ and $\{pk^*_{do}, sk^*_{do}\}$.

TagGen(SK, F). Before uploading a file F to decentralized storage network, data owner should process F using symmetric encryption algorithms to protect data privacy, such as AES, and erasure-correcting code [38] to reinforce data recoverability. Then, the file F is split into m blocks $\{f_1, f_2, \cdots f_m\}$ and each $2n$ blocks form a chunk. Thus, the number of chunks is $\lceil m/2n \rceil$ represented by d. The file is the form of $\{F_1, F_2, \cdots F_d\}$, and each chunk F_i can be denoted as $\{f_{i,1}, f_{i,2}, \cdots f_{i,2n}\}$. We use $\widehat{F}$ to denote the processed file.

It is noteworthy for data owner that the last chunk may need padding. For each F_i, the data owner generates a corresponding homomorphic authenticator utilizing the Pedersen-based polynomial commitment [22, 35]. An authenticator is computed as the following operations, where id is the file identifier randomly selected from $\mathbb{Z}_p$ and $H_1(id\|i)$ is the index information of chunk bound in the authenticator.

$$\begin{aligned}\sigma_i &= \left(\prod_{j=1}^{n} g_1^{f_{i,j}\alpha^{j-1}} \cdot \prod_{j=n+1}^{2n} h_1^{f_{i,j}\alpha^{j-n-1}} \cdot H_1(id\|i)\right)^{\upsilon} \\ &= \left(g_1^{M_{i1}(\alpha)} \cdot h_1^{M_{i2}(\alpha)} \cdot H_1(id\|i)\right)^{\upsilon}.\end{aligned} \tag{3}$$

Using the polynomial commitment based on Pedersen, we can commit two polynomials at one time without increasing the amount of calculation, which greatly alleviates the storage redundancy and I/O overhead. As shown in Equation (3), $M_{i1}(x)$ and $M_{i2}(x)$ are bound to the authenticator in the form of polynomial commitment, and their parameters are the first and latter half part of each chunk F_i. Besides, considering the computational cost for SNs in the self-auditing stage, the number of each chunk has an upper limit of $2n$.

FileDistribution(pk^*_{sni}, k). In order to ensure data recoverability, the data owner selects several nodes in the decentralized distributed network, and each node stores a part of the file. In detail, there are k storage nodes selected to store file chunks and accompanying authenticators, $\{F_i, \sigma_i\}_{i=1}^d$.

Each storage node SN_i should store metadata pair set $\{\{F_j\},\{\sigma_j\}\}_{j\in I}$ whose index set is computed as $I = \{H_2(pk^*_{sni}\|j) == i \bmod k\}_{j=1}^{d}$. Then, the data owner transfers the divided metadata pairs to each node in a secure channel and sends deposit to smart contract compiling in advance on the blockchain. After nodes receive the corresponding metadata, they also send deposit to the smart contract.

ChallGen$(pk^*_{sni}, t, \text{blockhash})$. Utilizing the information on the blockchain, SN_i generates a challenged set using the hash function. The height of the blockchain B_{init} is taken as the initial audit point where DO's deposit is sent to the blockchain. And each audit is generated for every B_{interval} blocks. Concretely, SN_i generates a challenged set $C_i = \{\{c_j, d_j\}_{j=1}^{t}, r_i\}$, where $c_j = H_2(\text{blockhash}\|pk^*_{sni}\|j)$, $d_j = H_2(\text{blockchain}\|c_j)$, and $r_i = H_2(\text{blockhash}\|pk^*_{sni})$.

Self-Auditing$(PK, \{C_i\}_{i=1}^{t}, \widehat{F}, \{\sigma_i\}_{i=1}^{d})$. Based on the file and authenticators, the node SN_i computes:

$$\sigma = \prod_{i=1}^{t} \sigma_{c_i}^{d_i}. \tag{4}$$

SN_i generates two polynomials, $P_1(x)$ and $P_2(x)$, which are linear combinations of t challenged chunks. Note that $F_i = \{f_{i,1},\cdots f_{i,2n}\}$.

$$\begin{aligned} P_1(x) &= \sum_{j=1}^{t} d_j f_{c_j,1}\cdots + \sum_{j=1}^{t} d_j f_{c_j,n-1}\cdot x^{n-1}, \\ P_2(x) &= \sum_{j=1}^{t} d_j f_{c_j,n+1}\cdots + \sum_{j=1}^{t} d_j f_{c_j,2n}\cdot x^{n-1}. \end{aligned} \tag{5}$$

Besides, computing quotient and remainder under the polynomial $(x - r_i)$, that is, $Q_i(x) = (P_i(x) - P_i(r_i))/x - r_i$, and we represent the coefficients vector of the quotient polynomial $Q_i(x)$ as $(w_{i,1}, w_{i2},\cdots,w_{i,n-2})$. In the end, SN_i produces:

$$\varphi = g_1^{Q_1(\sigma)} h_1^{Q_2(\sigma)} = \prod_{j=1}^{n-2} \left(g_1^{\alpha^j}\right)^{w_{1j}} \left(h_1^{\alpha^j}\right)^{w_{2j}}. \tag{6}$$

The form of proof is Proof_i=$\{\sigma, \varphi, P_1(r_i), P_2(r_i)\}$. Then, the node sends the Proof_i to the blockchain. After all nodes send proofs to the blockchain, every node verifies the correctness of other node's proofs through the following Equation (7). If each Proof_i is passed the verification equation, the node SN_i sends 1 to the smart contract; otherwise, 0. In the equation, χ, ϕ can, respectively, be acquired by calculating $\prod_{i=1}^{t} H(id\|c_i)^{d_i}$ and $g_1^{-P_1(r_i)} h_1^{-P_2(r_i)}$.

$$e(\sigma, g_2)e(\phi, \varepsilon) = e(\chi, \varepsilon)e(\varphi, \beta\cdot\varepsilon^{-r_i}). \tag{7}$$

The correctness of the above equation is proved as follows:

$$\begin{aligned} \text{LHS} &= e\left(g_1^{vP_1(\alpha)} h_1^{vP_2(\alpha)} \chi^v, g_2\right) e\left(g_1^{-P_1(r_i)} h_1^{-P_2(r_i)}, g_2^v\right) \\ &= e\left(g_1^{(P_1(\alpha)-P_1(r_i))} h_1^{(P_2(\alpha)-P_2(r_i))}, g_2^v\right) e(\chi^v, g_2) \\ &= e\left(g_1^{(\alpha-r_i)Q_1(\alpha)} h_1^{(\alpha-r_i)Q_2(\alpha)}, g_2^v\right) e(\chi, g_2^v) = \text{RHS}. \end{aligned} \tag{8}$$

Remark. Compared with Su et al. [27] in the self-auditing stage, our scheme never needs interaction between SNs. At the same time, we use blockchain information to generate the challenge set for each SN. Blockchain assists SNs in completing each audit without DO online. Therefore, we reduce the number of interactions and some computational overhead between SNs and DO.

4.3. Batch Verification. Our scheme can support batch auditing for improving the efficiency of data audits via polynomial commitment aggregation [37]. Consequently, our scheme can verify multiple proofs at one time. Concretely, when the last proof is transferred to the blockchain, every SN generates a random number $r = H_2(\text{blockhash})$. Based on the proofs $\{\text{Proof}_i\}_{i=1}^{k}$ on the blockchain, every SN generates the aggregation of the proofs as shown in Algorithm 1.

Finally, each SN sends the audit results to the smart contract. The smart contract maintains a counter for all storage nodes denoted the number of successful audits. After archiving the data stored in the decentralized network, SNs are rewarded or punished according to the number of counter.

In the end, we show that Algorithm 1 can ensure the correctness of batch verification. There have k proofs to audit at one time and the correctness of batch verification is shown as follows:

$$\begin{aligned} \text{LHS} &= \prod_{i=1}^{k} \left(e(\sigma_i, Z_i) e\left(g_1^{-p_1(r_i)} h_1^{-p_2(r_i)}, \widehat{Z}_i\right)\right)^{r^{i-1}} \\ &= \prod_{i=1}^{k} \left(e\left(g_1^{p_1(\alpha)-p_1(r_i)} h_1^{p_2(\alpha)-p_2(r_i)}, \widehat{Z}_i\right)\right)^{r^{i-1}} \\ &= \prod_{i=1}^{k} e\left(g_1^{Q_1(\alpha)} h_1^{Q_2(\alpha)}, Z_T\right)^{r^{i-1}} = \text{RHS}. \end{aligned} \tag{9}$$

5. Analysis of our Proposed Scheme

5.1. Security Analysis. In this section, we evaluate the security of our auditing scheme according to data privacy, storage correctness, and fairness listed in Section 3.2.

Theorem 1. *The scheme can guarantee that the correct proof can pass the verification, and the storage nodes cannot forge the authenticators and proofs when he does not maintain the entire data in the q-BSDH assumption.*

5.1.1. Data Privacy. We assume that the tools and cryptographic primitives used in our scheme are secure, such as the hash function, symmetric encryption algorithm, and polynomial commitment scheme. Therefore, we can ensure

Require:
The proofs shown on the blockchain:
$\{\text{Proof}_i = \sigma_i, \varphi_i, P_1(r_i), P_2(r_i)\}_{i=1}^{k}$;
The challenge r can compute by the information of blockchain, $r = H_2(\text{blockhash})$;
Ensure:
Result of integrity audit s;
1: Compute $\varphi = \prod_{i=1}^{k} \varphi_i^{r^{i-1}}$;
2: Compute $Z_i = g_2^{\Pi(\sigma - r_j)/(\sigma - r_i)}$, $\widehat{Z}_i = g_2^{v\Pi(\sigma - r_j)/(\sigma - r_i)}$;
3: Compute $Z_T = g_2^{v\Pi(\sigma - r_i)}$;
4: Compute A as: $\prod_{i=1}^{k} \left(e(\sigma_i, Z_i) e(g_1^{-p_1(r_i)} h^{-p_2(r_i)}, \widehat{Z}_i) e(\chi_i^{-1}, \widehat{Z}_i)\right)^{r^{i-1}}$
5: Compute $B = \mathrm{e}(\varphi, Z_T)$;
6: **If** $A = B$ **then**
7: Set $s = 1$;
8: **else**
9: Set $s = 0$;
10: **end if**
11: **return** s;

Algorithm 1: SN batch verification.

the privacy of the DO's data by using symmetric encryption algorithm before uploading data to the decentralized storage network.

5.1.2. Storage Correctness and Fairness. In Equation (8) and Equation (9), we first prove that the honest SN can always pass the integrity verification. Then, we give a proof sketch that the authenticator generated by DO and the proof generated by SN are unforgeable.

According to the description in [35], the Pedersen-based polynomial commitment scheme is security provided the t-SDH assumption in group $\mathbb{G}_1$. Therefore, if an existed probabilistic polynomial time adversary $\mathscr{A}$ can forge an authenticator, $\mathscr{A}$ can construct an algorithm to efficiently deal with the t-SDH problem. Specifically, assume that $\mathscr{A}$ can forge $f_1(x)$ and $g_1(x)$ such as $g^{f_1(\alpha)} h^{g_1(\alpha)} = g^{f_2(\alpha)} h^{g_2(\alpha)}$, where $f_1(x)$, $g_1(x)$, $f_2(x)$, and $g_2(x)$ are known to $\mathscr{A}$. $\mathscr{A}$ can construct polynomials $f(x) = f_1(x) - f_2(x)$ and $g(x) = g_1(x) - g_2(x)$ and gain $g^{f(x)} h^{g(x)}$. Therefore, to factor $f(x)$ and $g(x)$, $\mathscr{A}$ can achieve the private key α and break the t-SDH assumption in the system security parameters.

We show that malicious SN cannot generate valid proof if the entire data is never honestly stored. Given two valid proof responses $(\sigma_1, \varphi_1, P_1(r_1), P_2(r_1))$ and $(\sigma_2, \varphi_2, P_1(r_2), P_2(r_2))$. An adversary $\mathscr{A}$ is possible to extract the knowledge of linear combination of original data chunks with overwhelming probability unless $\mathscr{A}$ can deal with CDH and q-BSDH problems. Please refer to [3, 22, 39] for more details.

Moreover, the scheme generates a probabilistic proof of data possession like previous work. To guarantee a high confidence level is significant for DO. The probabilistic analysis given in [2] shows the relationship between the storage confidence level and the number of challenged chunks. Concretely, if 1% of data chunks have been tampered, only 300 chunks can give DO storage guarantee level of 95%. In the decentralized storage network, we think this number of challenges is sufficient to protect DO interest.

Remarks on Fairness. We further analyze the fairness in our scheme. First of all, we consider that the DO generates the incorrect metadata to gain more benefit. After the SN receives this erroneous metadata, it can implement the audit protocol locally in advance to ensure the correctness of the metadata with a tremendous probability. If the SN finds the wrong metadata, it can stop this transaction at negligible cost. Then, we consider that the malicious SN may generate wrong proof to gain more interest. The most honest nodes can send the malicious node's blockchain address to the smart contract when the audit fails. Then, the smart contract can transfer the deposit of the malicious node to other participants.

5.2. Comparison. As shown in Table 2, we compare four data auditing schemes, which consists of Yuan's scheme [22], Du's scheme [9], Su's scheme [27], and our scheme, in terms of audit mode, no interactions between SNs, data owner offline, decentralized storage, and batch auditing. Firstly, compared with Yuan's scheme, other schemes remove TPA from the traditional system model, which defends against single-point-of-failure and collusion between storage providers and TPA. Secondly, only Su's scheme has $4k$ interactions between SNs in the self-auditing stage because each node needs to obtain other nodes' proofs to complete each auditing task where the parameter k is the number of storage nodes.

Finally, Su's scheme and our scheme support decentralized storage and batch auditing, improving data recoverability and reducing the computation overhead for SNs.

Both Su's scheme and our scheme use self-auditing to remove TPA in traditional system model. However, our scheme uses blockchain information to generate the challenged set for each SN. Thus, data owners should never

Table 2: Comparison of data auditing schemes.

	Audit mode	No interactions between SNs	Data owner offline	Decentralized storage	Batch auditing
Scheme [22]	TPA	Yes	Yes	No	No
Scheme [9]	Smart contract	Yes	Yes	No	No
Scheme [27]	Self-auditing	No	No	Yes	Yes
Our scheme	Self-auditing	Yes	Yes	Yes	Yes

Table 3: Performance comparison.

	File information	Preprocess		Proof generation		Verification	Multiple proof verification
	Size	Time	Tag size	Time	Size	Time	Time
Scheme [9]	1 GB	86 s	13 MB	45 ms	288 byte	45 ms	88 ms
Our scheme	1 GB	82 s	6 MB	61 ms	320 byte	46 ms	51 ms

Note that we compare with scheme [9] in the parameter n to 150 and the size of challenged set t to 240.

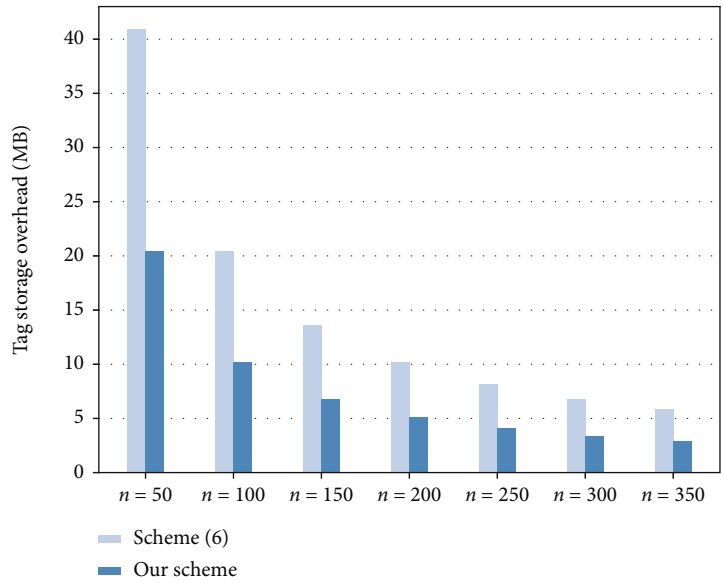

Figure 3: The size of authenticators of 1 GB data.

online to challenge each SN, and each SN should only interact with blockchain without amount communication overhead. However, Su's scheme should have $4k$ interactions between SNs which is unfeasible when k becomes larger.

6. Performance Evaluation

This section gives a performance evaluation of our auditing scheme in terms of off-chain storage and computational overhead and on-chain gas fee overhead. Besides, we mainly compare the experimental result with Du's scheme [9] which is the first auditing framework to consider on-chain privacy and efficiency.

6.1. Implementation and Experiment Setup. We leverage the Golang language to implement our off-chain scheme by the BN256 curve [40] and the KZG [41] library where the secure parameter is 256 bit. BN256 curve library implements the elliptic-curve-related operations with Golang language ($(|p| = |\mathbb{G}_1| = 256\text{bit}, |\mathbb{G}_2| = 512\text{bit})$. KZG library implements addition, subtraction, multiplication, and division of polynomials with Golang language. On the part of the blockchain, we use the Ethereum test network Reposten and utilize the Solidity and Remix-IDE tool to employ our smart contract.

In order to simulate the decentralized storage network, we use three virtual machines as our storage nodes with Intel (R) Core (TM) i5-9500 CPU 3.00 GHz, 4 GB RAM and 20 GB SSD disks running on Ubuntu 18.04 LTS. The data owner uses a Desktop PC with windows 10 (AMD Ryzen5 4600 U CPU 2.1 GHz 16 GB RAM). Besides, we use the same configuration and parameters to implement the Du's scheme [9]. We set p as a 256 bit large prime number, the stored file size to 1 GB, and the number of challenged chunks from 240 to 440. The evaluation results are the average of 10 experiments.

6.2. Off-Chain Overhead. In the off-chain part, we test the authenticators' generation time, authenticators storage overhead, and proof generation time as shown in Table 3. To

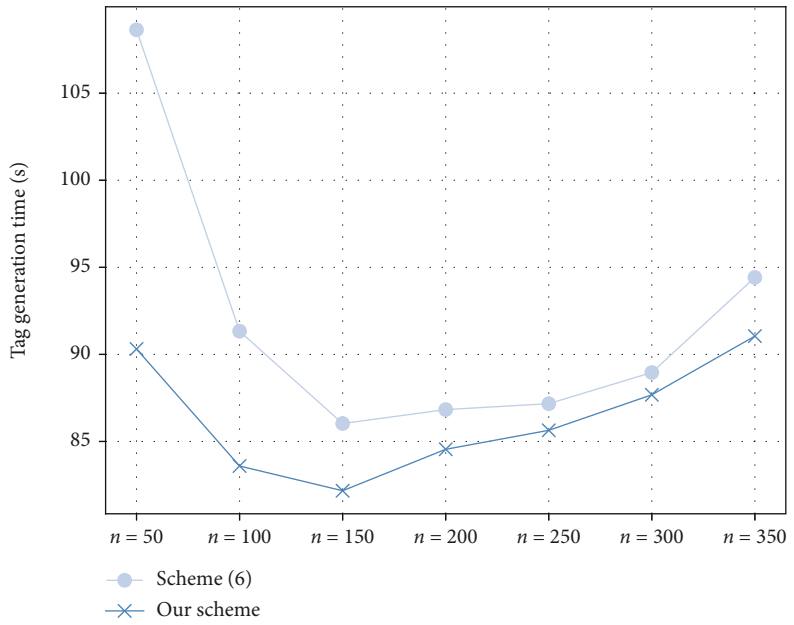

Figure 4: The computational time for DO to process 1 GB data.

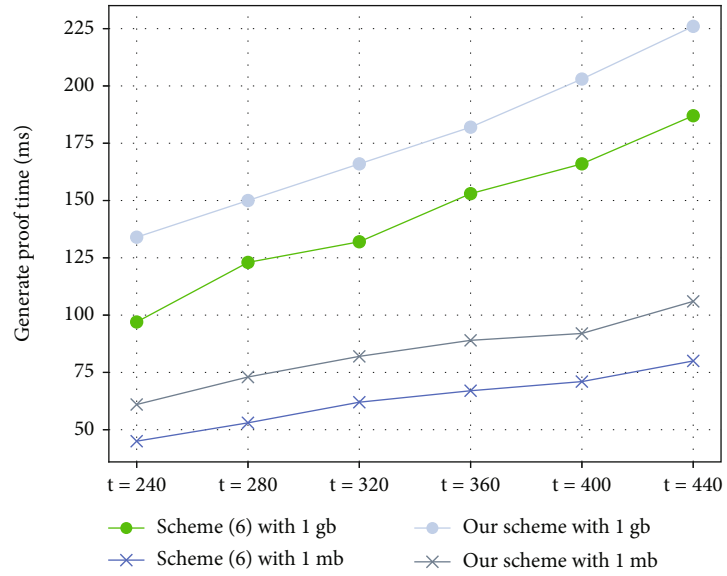

Figure 5: The proof generation time.

simulate data owner in the preprocessing stage, we use four Golang coroutines to process a 1 GB file size with the parameter n from 50 to 350. Note that our authenticators' generation time includes other factors such as key pairs' generation, I/O overhead, and polynomial coefficient transformation of data chunks.

The parameter n is negatively correlated with the number of chunks and authenticators. Thus, as illustrated in Figure 3, we greatly decrease the number of authenticators which is always half of Du's scheme [9] due to the use of Pedersen-based polynomial commitment. As shown in Figure 4, we slightly decrease the authenticators' generation time due to less frequency to access the original file. Therefore, we spend less time on I/O overhead with the increasing of parameter n compared with Du's scheme. However, $n-1$ is the order of two committed polynomials in the TagGen stage. The computational overhead for the polynomials becomes more expensive with the parameter n increases. We can see that the case of $n = 150$ is the least time for generating the auditing authenticators.

Then, we test the overhead of proof generation related to the number of the challenged chunks from 240 to 440. In the proof generation phase, our scheme needs to deal with an extrapolynomial with the order of $n-1$. Therefore, as shown

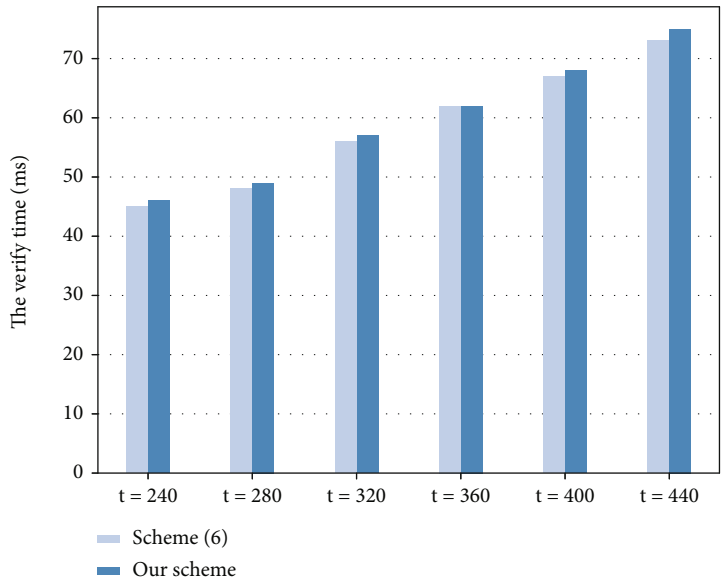

Figure 6: The single verification time in the parameter $n = 150$.

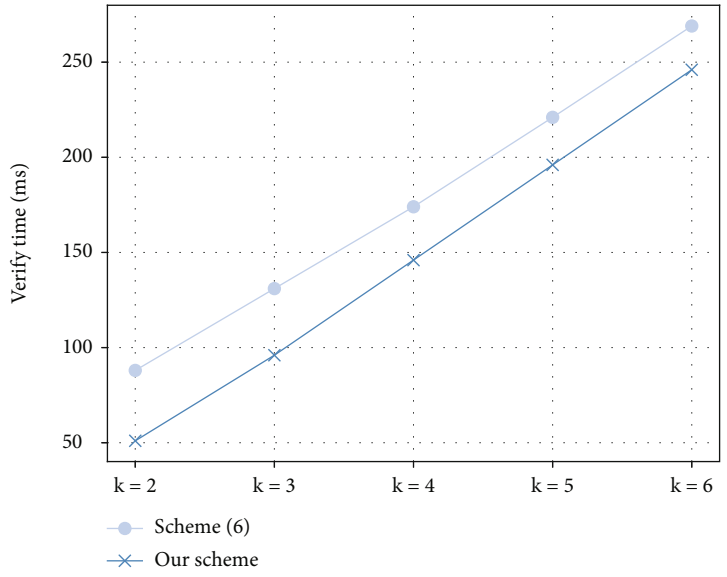

Figure 7: Multiple proof verification time.

Table 4: Gas consumption comparison (million).

	PK storage	Chal-generation	Proof storage	Proof verification
Scheme [9]	0.16	0.07	0.029	0.6
Our scheme	0.30	0	0.031	0

Note that we compare the gas consumption with scheme [9] in the parameter n to 50 and the size of challenged set t to 300.

in Figure 5, compared with Du's scheme, our scheme is about 20-30% slower in generating proof. Moreover, we can discover that the size of file dramatically affects the proof generation time, mainly dominated by the I/O cost. Due to the smaller authenticators' redundancy, our scheme has less I/O overhead. In the case of 1 MB, which can ignore the cost of I/O, we are about 30% slower than the comparison scheme. And for the case of 1 GB, we are about 20% slower than the comparison scheme. Thus, our scheme has the advantage of processing more larger file.

Then, setting the parameter n to 150, we evaluate the verification time by increasing the challenged chunks from 240 to 440. As shown in Figure 6, the time consumption increases linearly with the size of the challenged chunks. Compared with Du's scheme, our scheme requires additional overhead to calculate the multiplication and addition on elliptic curve $\mathbb{G}_1$, but this time can be ignored. As shown in the case of chunks of 240, the verification time of the two schemes is almost equal, about 45 ms.

Lastly, we test the time cost of batch verification with the increasing number of storage nodes. We set the number of blocks to 150, challenged chunks to 240, and storage nodes from 2 to 7. As shown in Figure 7, the test results of our batch verification are compared with Du's k-time single verification. In theory, we can decrease $k-1$ bilinear pairing operations, and other calculation overhead is unchanged when k is small. It can be seen from the figure that our batch verification scheme can improve 30% verification efficiency.

6.3. On-Chain Overhead. In the decentralized system model based on blockchain, the on-chain cost is concerned by the participants. We use a smart contract to manage all participants who use our scheme. All participants only need to interact with this smart contract rather than create their own smart contracts. The proofs and counters of SNs and the public keys of DOs are all stored in this smart contract. We use the Ethereum test network, Reposten, to deploy our smart contract written by Solidity. There are two main function in our smart contract, storage and auditCount.

As shown in Table 4, we compare the on-chain gas overhead in terms of the storage of public key and proofs, challenge generation, and proof verification.

After the setup phase, our scheme should send PK to the blockchain. Note that the size of a $\mathbb{G}_1$ element is 64 bytes and $\mathbb{G}_2$ is 128 bytes. To save gas consumption, we only send x -axis coordinates on the elliptic curve to the blockchain, and more 1 bit records the positive or negative information of the elliptic curve points. Through computing the y-axis coordinates on the off-chain, SNs can obtain public keys. Compared with Du' scheme, our scheme only adds $n-2$ group elements of $\mathbb{G}_1$ and $2k$ elements on $\mathbb{G}_2$ groups to improve the size of the authenticator and introduce the batch auditing. Thus, the consumption of PK storage of our scheme is 0.3 million, which is nearly a double increase compared with Du's scheme. However, note that this process is one-time storage cost for the whole auditing duration. Data owners can use these public keys in the future. Moreover, we compare the storage overhead of the proof in the blockchain. Compared with Du's scheme, our proof only has an extra 256 bit large number $P_2(r_i)$. We increase about 2000 gas consumption in each audit for the test in the Reposten.

However, compared with Du's scheme in the challenge generation and proof verification phase, our scheme never requires any overhead on the blockchain. We are only required to maintain a counter to record the number of successful audits without the amount of computation in the smart contract. To sum up, our scheme saves approximately 60% gas consumption.

7. Conclusion

In this paper, we proposed a blockchain-based self-auditing scheme with batch verification in a decentralized framework. Firstly, different from previous works, our scheme removes TPA through the interaction between storage nodes and blockchain to achieve self-auditing. The recoverability of data can be guaranteed due to the distribution to storage nodes. Secondly, using the Pedersen-based polynomial commitment to generate the authenticators, our scheme decreases the computational overhead for DO and the storage overhead for SNs. Moreover, the batch verification algorithm improves the verification efficiency by aggregating multiple polynomials and points. Lastly, security analysis and experiments show that our scheme achieves the security goals and is efficient and feasible to deploy in the practice blockchain environment.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (no. 62072357), the Key Research and Development Program of Shaanxi (nos. 2022KWZ-01 and 2020ZDLGY08-03), the Shandong Provincial Key Research and Development Program of China (no. 2019JZZY020129), and the Fundamental Research Funds for the Central Universities (nos. JB211503 and YJS2212).

References

[1] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1036–1048, 2022.

[2] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598–609, Alexandria Virginia USA, 2007.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90–107, Springer, 2008.

[4] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[5] ethereum, "The golang version of etherum client," 2021, https://github.com/ethereum/go-ethereum.

[6] A. Yakovenko, "Solana: a blockchain network focused on fast transactions and high throughput," 2019, https://solana.com.

[7] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Information Sciences*, vol. 541, pp. 409–425, 2020.

[8] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020.

[9] Y. Du, H. Duan, A. Zhou, C. Wang, M. H. Au, and Q. Wang, "Towards privacy-assured and lightweight on-chain auditing of decentralized storage," in *2020 IEEE 40th International*

Conference on Distributed Computing Systems (ICDCS), pp. 201–211, Singapore, Singapore, 2020.

[10] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 584–597, Alexandria Virginia USA, 2007.

[11] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *International Conference on the Theory and Application of Cryptology and information Security*, pp. 514–532, Springer, 2001.

[12] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "Mr-pdp: multiplereplica provable data possession," in *2008 the 28th international conference on distributed computing systems*, pp. 411–420, Beijing, China, 2008.

[13] H. Kai, H. Chuanhe, W. Jinhai et al., "An efficient public batch auditing protocol for data security in multi-cloud storage," in *2013 8th China Grid Annual Conference*, pp. 51–56, Los Alamitos, CA, USA, 2013.

[14] J. Chang, B. Shao, Y. Ji, and G. Bian, "Efficient identity-based provable multi-copy data possession in multi-cloud storage, revisited," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2723–2727, 2020.

[15] I. Damgård, C. Ganesh, and C. Orlandi, "Proofs of replicated storage without timing assumptions," in *Annual International Cryptology Conference*, pp. 355–380, Springer, 2019.

[16] F. Armknecht, L. Barman, J.-M. Bohli, and G. O. Karame, "Mirror: enabling proofs of data replication and retrievability in the cloud," in *25th USENIX security symposium (USENIX security 16)*, pp. 1051–1068, Austin, TX, 2016.

[17] I. Leontiadis and R. Curtmola, "Secure storage with replication and transparent deduplication," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pp. 13–23, Tempe AZ USA, 2018.

[18] J. Ning, J. Chen, K. Liang, J. K. Liu, C. Su, and Q. Wu, "Efficient encrypted data search with expressive queries and flexible update," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1619–1633, 2020.

[19] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 212–225, 2021.

[20] X. Zhang, J. Zhao, C. Xu, H. Li, H. Wang, and Y. Zhang, "CIPPPA: conditional identity privacy-preserving public auditing for cloud-based wbans against malicious auditors," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1362–1375, 2021.

[21] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3293–3303, 2015.

[22] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability and constant communication cost in cloud," in *Proceedings of the 2013 international workshop on Security in cloud computing*, pp. 19–26, Hangzhou China, 2013.

[23] H. Wang, Q. Wang, and D. He, "Blockchain-based private provable data possession," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2379–2389, 2021.

[24] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 923–937, 2021.

[25] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

[26] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1421–1432, 2021.

[27] Y. Su, Y. Li, B. Yang, and Y. Ding, "Decentralized self-auditing scheme with errors localization for multi-cloud storage," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.

[28] Y. Li, Y. Yu, R. Chen, X. Du, and M. Guizani, "IntegrityChain: provable data possession for decentralized storage," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1205–1217, 2020.

[29] D. Francati, G. Ateniese, A. Faye et al., "Audita: a blockchain-based auditing framework for off-chain storage," in *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, pp. 5–10, Virtual Event Hong Kong, 2021.

[30] D. Chen, H. Yuan, S. Hu, Q. Wang, and C. Wang, "Bossa: a decentralized system for proofs of data retrievability and replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 786–798, 2021.

[31] S. fund, "Swarm: a decentralised storage and communication system for a sovereign digital society," https://www.ethswarm.org.

[32] S. Labs, "A decentralized cloud storage network framework," 2018, https://www.storj.io.

[33] D. Vorick and L. Champine, "Sia: simple decentralized storage," *Retrieved May*, vol. 8, p. 2018, 2014.

[34] P. Labs, "Filecoin: a decentralized storage network," 2017, https://filecoin.io/.

[35] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 177–194, Springer, 2010.

[36] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Annual International Cryptology Conference*, pp. 129–140, Springer, 1992.

[37] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, *Efficient Polynomial Commitment Schemes for Multiple Points and Polynomials*, Cryptology ePrint Archive, 2020.

[38] J. Bloemer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, *An Xor-Based Erasure-Resilient Coding Scheme*, International Computer Science Institute (ICSI) Technical Report, 1995.

[39] D. Boneh and X. Boyen, "Short signatures without random oracles," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 56–73, Springer, 2004.

[40] Cloudflare, "Package bn256 implements a particular bilinear group," 2019, https://github.com/ethereum/go-ethereum/tree/master/crypto/bn256/cloudflare.

[41] Arnaucube, "The kzg polynomials commitment," 2021, https://github.com/arnaucube/kzg-commitments-study.