Hindawi
Security and Communication Networks
Volume 2022, Article ID 6268449, 14 pages
https://doi.org/10.1155/2022/6268449

Research Article

A Noninteractive Multireplica Provable Data Possession Scheme Based on Smart Contract

Zhengwen Li,[1] **Yang Xin**,[1] **De Zhao**,[2] **and Yixian Yang**[1]

[1]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*School of Information Engineering, Beijing Institute of Graphic Communication, Beijing 102600, China*

Correspondence should be addressed to Zhengwen Li; lizhengwen@bupt.edu.cn

Received 19 November 2021; Revised 25 January 2022; Accepted 8 March 2022; Published 6 April 2022

Academic Editor: Xin-Yi Huang

With the explosive growth of data, cloud storage has become a widely used storage method. To protect the integrity and availability of data in cloud storage systems, multireplica provable data possession has gradually become a research hotspot. This paper uses smart contracts to replace traditional third-party auditor (TPA) and proposes a noninteractive multireplica provable data possession scheme based on smart contracts, making the verification process public, immutable, traceable, and able to be carried out periodically and automatically. This paper introduces the concept of noninteractivity to reduce the transaction fees caused by the frequent operation of blockchain in the verification process. By stipulating payment rules in the smart contract, we can ensure the fairness of all parties. Finally, we give the correctness proof of the scheme and the security proof in the random oracle model, comparing it with other schemes and verifying the practicability of our scheme through experiments.

1. Introduction

In recent years, with the continuous development of the Internet of Things, big data, artificial intelligence, mobile Internet, and other fields, the amount of data generated by people has increased explosively. According to IDC [1], the total amount of data in the world will increase from 33 ZB in 2018 to 175 ZB in 2025, and the data will become a precious strategic resource. Cloud storage has gradually become the data storage trend due to its low cost, flexible scalability, and anytime and anywhere access. The most popular products are Amazon S3, Google Drive, Microsoft Azure, Dropbox, Alibaba Cloud, etc.

Generally, after uploading local data to the cloud server using cloud storage services, users will delete the original data to save local storage resources. Due to the separation of cloud storage data ownership and physical control, users cannot timely understand the actual storage status of data, which makes the availability and integrity of data one of the most concerning issues of cloud storage security for users.

Data availability means that users can get data in time when they need it and recover the original data when there is a certain degree of error in the data. To ensure data availability, multireplica and erasure code are two widely used technologies. Multireplica technology usually stores multiple replicas on multiple servers. If a replica of data is damaged, it can be recovered using replicas of other data centers. Erasure code is a coding technology, which uses redundant blocks to provide fault tolerance. When part of the data is damaged, it can be reconstructed by coding. Compared with erasure code, multireplica technology uses more storage space, but its implementation is more straightforward and consumes less computing resources, so it is more widely used.

Data integrity means that specific data remain completely unchanged during storage or transmission. To ensure data integrity, Provable Data Possession (PDP) and Proofs of Retrievability (POR) are two widely used methods. PDP is used mainly to complete data integrity verification quickly, and POR is used to ensure data integrity due to its ability to recover data. It consumes additional computing resources and storage space.

Cloud storage service providers (CSP) are not entirely credible. User data may be damaged and unavailable due to power interruption, hacker attacks, and software and hardware failures, and even some CSPs deliberately tamper, destroy, and delete user data for some purpose. To avoid downloading data before finding that the data are unavailable, users should periodically check the data integrity in the CSP. Combined with the current situation that CSP has widely adopted multireplica technology, doing multireplica provable data possession safely and efficiently has become a research hotspot in recent years.

1.1. Related Work. Ateniese et al. [2] first proposed the PDP scheme, which obtains the probability of data integrally possessed by the server through random sampling of data blocks, allowing users to check whether the server has stored the entire data without downloading all data. It uses the homomorphic verification tag based on RSA to reduce the computational overhead and improve the efficiency of integrity verification. This scheme is also the first to support public verification, which can meet the needs of third-party verification. Around the same time, Juels and Kaliski [3] proposed the POR scheme, which is based on the sentinel mechanism and can restore damaged data while providing integrity verification. Since the number of sentinels is fixed and the verification consumes several sentinels each time, the scheme has finite verification times. In addition, this paper presents a formal security definition of integrity verification for the first time, which is instructive for follow-up research. Shacham and Waters [4] proposed two POR schemes based on BLS signature and pseudorandom functions. They presented complete proofs of security of the two schemes under the random oracle model and the standard model through the interactive analysis of a series of games. Wang et al. [5] proposed a PDP scheme based on the BLS signature, which supports public verification and dynamic data update by constructing Merkle hash trees of the data block tag authenticator. To fully ensure data security and conserve the computational resources of users, Wang [6] introduced the TPA to complete the verification work and propose a public verification scheme supporting privacy preserving by combining homomorphic linear authenticator and random masking technique, which can batch process multiple verification tasks.

To satisfy users' demands for data availability, CSP duplicates the data into corresponding replicas and stores multiple replicas on multiple servers. For multireplica provable data possession, Curtmola et al. [7] proposed an MR-PDP scheme to reduce the overhead of integrity verification of all copies to roughly the same as a single copy. Unfortunately, this scheme only supports private verification. Hao and Yu [8] proposed a multireplica remote data possession check protocol with public verifiability by combining a homomorphic verification tag and BLS signatures. Wei [9] proposed an efficient dynamic replicated data possession verification scheme, which uses the fully homomorphic encryption (FHE) algorithm to generate multiple replicas and resist forgery, replacement, and replay attacks. Ya-Xing [10] proposed a new multiuser and multiple-replica provable data possession scheme. The scheme adopts random mask technology to process ciphertext to ensure data privacy. It adopts a multibranch authentication tree to improve the efficiency of data block signature, which can support dynamic data update operation and batch audit. Peng et al. [11] proposed an identity-based multiple-replica data integrity checking scheme (EDID-MRPDP), introducing a new Homomorphic Verifiable Tag (HVT) structure and a new Compressed Authentication Array (CAA) data structure, which can simultaneously and efficiently conduct batch authentication for multiple owners and cloud servers. Yu et al. [12] proposed a dynamic multiple-replica auditing scheme, which can simultaneously verify the integrity and geographic location of the replica data of cloud users by introducing an Indexed Merkle Hash Tree (IMHT), and the problem of the excessive overhead of the existing Merkle hash tree can be reduced.

Most of the above integrity verification schemes assume that TPA is credible, which is bold and dangerous. If the auditor colluded with the CSP or the attacker, the provable data integrity provided by the auditor would become unreliable [13–17]. Meanwhile, TPA is also faced with a single point of failure and performance limitations. Fortunately, the emergence of blockchain technology provides a new way to solve these problems because the essence of blockchain technology is a mutual trust mechanism based on mathematical algorithms. In addition, blockchain has the characteristics of decentralization, openness, transparency, tamperproof, and traceability, which coincide with the requirements of data integrity audit. Nowadays, more and more scholars have begun to combine blockchain technology to research provable data integrity.

Some schemes [18–20] only take advantage of the openness, transparency, and tamper-proof characteristics of the blockchain to store verification logs on the blockchain but do not eliminate the threat of malicious TPA. Huang et al. [21] proposed a collaborative auditing blockchain framework for cloud data storage by using all consensus nodes substituting the single third-party auditor to execute auditing delegations and record them permanently, but not for multiple replicas. Xu [22] proposed a decentralized and arbitrable data auditing scheme based on blockchain. It mainly uses the communicative hash technique to randomly verify the integrity of a group of data blocks to probabilistically verify the integrity of all data, and it completes the information interaction in the verification process through blockchain transactions. It uses smart contracts to realize the adjudication mechanism without TPA. However, the scheme is too idealized, almost every data block needs to participate in the verification, and the interaction process will produce a large number of blockchain transaction costs, which makes the scheme very impractical. Chen et al. [23] proposed the first decentralized system BOSSA for proofs of data retrievability and replication. Since the blockchain cannot actively issue challenges and reacts based on received transactions, this paper proposes a time-restricted proof forcing the cloud to prove data availability. In addition, the scheme is aimed at the decentralized storage network, where other nodes store replicas, so the replicas must be encoded and encrypted to ensure privacy and reliability. Fan et al.

[24] used a smart contract to replace TPA and proposed a decentralized audit scheme on Ethereum called Dredas; anyone can obtain audit results from Ethereum without worrying about semihonest TPA. This solution uses a smart contract and ether to propose a deposit mechanism to pay audit fees and punish malicious behavior. At the same time, the solution also supports batch audit and dynamic data audit. However, the scheme does not consider the case of multiple replicas, and a large amount of information needs to be stored in the contract in the audit process, which has the problem of high interaction cost. Wang et al. [25] used blockchain to replace TPA and designed a blockchain-based fair payment smart contract for a public audit of cloud storage. This contract ensures that CSP needs to submit provable data possession termly. To reduce the number of times of interactions during the execution of the contract, the concept of noninteractive provable data possession was first proposed. Unfortunately, multireplica is not considered. Li et al. [26] proposed a decentralized storage framework supporting provable data possession based on blockchain—IntegrityChain, which can simultaneously protect data confidentiality, integrity, and availability by using pseudorandom function and multireplica technology. However, all interactions in this scheme are completed through transactions of blockchain, and gas is consumed in each step, so the transaction cost is significantly increased. Chen et al. [27] proposed a decentralized outsourcing storage system that supports dynamic provable data possession based on the blockchain. The applicable scenario is P2P storage network. All storage and audit behaviors will generate transactions, and then blockchain is used to record all transactions. The scheme utilizes smart contract to support public verification, ensures fairness of all parties by deposit mechanism, and takes advantage of an authenticated data structure (ADS) called rank-based Merkle hash tree to support updating operations. However, the scheme is not lightweight enough, and additional Merkle tree structure and auxiliary verification information need to be stored in the transaction, which has a great burden on the operation of blockchain.

Existing schemes do not fully account for the transaction fees on the blockchain, which would be higher if the data were to be manipulated in a complex manner. In addition, the storage capacity of each block is so small that it is impossible to store large amounts of data or complex data structures in practice. Therefore, considering the limited storage capacity of blocks and the high transaction fees caused by frequent interactions, we improved the multireplica provable data possession protocol based on the BLS signature and homomorphic authentication tag and proposed a noninteractive lightweight scheme combined with a smart contract.

1.2. Our Contribution. Our contributions are summarized as follows.

(1) A noninteractive multireplica provable data possession protocol NI-MR-PDP is designed to support public verification, batch processing, and privacy preserving; all parties in the system do not need to carry out challenge-response interaction. A series of games are constructed for interactive analysis to prove that the protocol is safe in the random oracle model.

(2) A noninteractive multireplica provable data possession scheme based on smart contracts is proposed. Deploying smart contracts on blockchain to eliminate the dependence on untrusted TPA can automatically verify data integrity openly, transparently, and periodically. According to the content of the smart contract, the rights and obligations of the participating parties are stipulated. Once the conditional contract is triggered, automatic execution of the contract can protect the legitimate rights of all parties and reduce the settlement cost of disputes. After deploying the contract, the parties in the system do not need to interact, which helps the consensus nodes in the blockchain to efficiently implement the smart contract.

(3) A series of games are constructed for interactive analysis to prove that the scheme is safe under the random oracle model. Experiments show that the scheme is practical and has good efficiency.

1.3. Organization. The rest of the paper is organized as follows. Section 2 recalls some preliminaries used in our scheme. Section 3 defines the model of our scheme, gives the formal definition, and presents the concrete construction. Section 4 provides the security proof of the protocol. Section 5 evaluates the performance of our scheme. Finally, we give a conclusion in Section 6.

2. Preliminaries

2.1. Bilinear Map. Let G_1, G_2, and G_T be multiplicative cyclic groups of prime order p, g_1 a generator of G_1, and g_2 a generator of G_2. A bilinear map e: $G_1 \times G_2 \longrightarrow G_T$ has the following properties.

(1) Bilinear: $\forall u \in G_1$, $v \in G_2$ and $\forall a, b \in Z_p$, there is $e(u^a, v^b) = e(u, v)^{ab}$

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

(3) Computable: $\forall u \in G_1, v \in G_2$, and there is an efficient algorithm to calculate $e(u, v)$

2.2. BLS Signature. Dan Boneh [28] proposed the BLS signature scheme, which uses bilinear mapping to verify the elements in the elliptic curve group. The core idea is to verify the correctness of the digital signature while protecting the user's private key from being leaked. The signature length of BLS is shortened to 160 bits, which is shorter than a typical signature at the same security level.

Let the signature algorithm be based on bilinear mapping e: $G_1 \times G_2 \longrightarrow G_T$, where G_1, G_2, and G_T are multiplicative cyclic groups of prime order p, g_1 is a generator of G_1, and g_2 is a generator of G_2.

BLS signature algorithm includes three algorithms: key generation algorithm, signature algorithm, and verification algorithm. The specific description is as follows.

(1) SKg: it is used to generate a pair of the public and secret keys of the signature scheme. The user randomly selects a value $x \in Z_p$ as the secret key, and the corresponding public key is $g_2^x \in G_2$.

(2) SSing: it is used to complete the signature of the message. Given a secret key x and message $m \in \{0,1\}^*$, compute the hash of the message $h = H(m), h \in G_1$ and output the signature $\sigma = h^x, \sigma \in G_1$.

(3) SVerify: it is used to verify the validity of the signature. Given a message m, signature σ, and public key g_2^x, check whether $e(\sigma, g) = e(h, g_2^x)$ holds. If it holds, the signature is valid. Otherwise, it is invalid.

2.3. Blockchain and Smart Contract. In 2008, Satoshi Nakamoto [29] proposed the concept of bitcoin, and blockchain as the core technology of bitcoin was proposed for the first time. Blockchain is essentially a chained data structure that combines data blocks in chronological order and is a tamper-proof and unforgeable distributed ledger guaranteed by cryptography.

In the blockchain, data are permanently stored in blocks, and blocks are generated one by one in chronological order and connected into a chain. As shown in Figure 1, each block contains a block header and a block body. The block header contains the previous block's hash value, version number, random value, timestamp, Merkle root hash, and difficulty value. The block body contains all transaction information generated during the block creation process. Each block in the blockchain is identified by a hash value obtained by the secondary SHA256 hash calculation of the block header. Each block can find its previous block by the previous block hash value contained in its block header. Any change to a block on the blockchain will lead to a series of changes in subsequent blocks. Distributed nodes synchronously update the hash chain by running a consensus protocol. Therefore, blockchain has the characteristics of decentralization, transparency, openness, tamper-resistant, and traceability.

In 1997, the smart contract was formally proposed by Nick Szabo [30], which is an electronic quantitative trading protocol for contract terms in reality. The essence of a smart contract is a piece of code running on the blockchain. The logic of the code defines the content of the smart contract. After the smart contract is successfully deployed, once the agreed rules are met, the contract content can be automatically executed without the participation of intermediaries, and no one can prevent it from running. We can say that blockchain provides a trusted execution environment for smart contracts, and smart contracts extend blockchain's application. The smart contract has been applied in many fields, such as electronic voting [31] and insurance [32], and has broad prospects.

3. Our Scheme

3.1. System Model. The system model of a noninteractive multireplica provable data possession scheme includes three entities: data owner, cloud storage service provider, and verifier (see Figure 2).

(1) Data Owner (DO): cloud storage service users choose to pay a certain fee to store their data in remote servers of the cloud storage service provider to save local storage costs and use data flexibly and conveniently.

(2) Cloud Storage Service Provider (CSP): it is composed of multiple replica servers, which adopt multireplica technology to improve data availability and provide computing resources, storage resources, and network bandwidth resources for DO. CSP needs to verify the data integrity of multiple replicas periodically. If the verification fails, it will compensate for a certain fee to DO.

(3) Verifier: it is the proof verification algorithm executor; because of the public verification of the scheme, theoretically all members of the blockchain can act as verifiers, usually by third-party miners. The verifier obtains a certain reward by executing the smart contract deployed on the blockchain.

3.2. Formal Definition. The scheme is divided into two phases: the setup and audit phases. It consists of five algorithms: KeyGen, ReplicaGen, TagGen, ProofGen, and ProofVerify. Each algorithm is formally defined as follows.

(1) $\mathrm{Key\,Gen}(1^\lambda) \longrightarrow (pk, sk)$: key generation algorithm is run by DO. The algorithm's input is a security parameter λ, and the output is public key pk and secret key sk.

(2) $\mathrm{Replica\,Gen}(F) \longrightarrow \{F_d\}$: replica generation algorithm is run by DO. The algorithm's input is ciphertext F, and the output is t different replicas $\{F_d\}, 1 \le d \le t$.

(3) $\mathrm{Tag\,Gen}(sk, F_d) \longrightarrow (\tau_d, \psi)$: tag generation algorithm is run by DO. The algorithm's input is the secret key sk and replica F_d, and the output is the tag of replica τ_d and the tag set of blocks ψ.

(4) $\mathrm{Proof\,Gen}(\theta, F_d, \tau_d, \psi) \longrightarrow P_d$: proof generation algorithm is run by CSP. The algorithm's input is public state information θ, replica F_d, the tag of replica τ_d, and the tag set of blocks ψ, and the output is proof $P_d, 1 \le d \le t$.

(5) $\mathrm{Proof\,Verify}(pk, \theta, P) \longrightarrow (\mathrm{SUCCESS}, \mathrm{FALSE})$: proof verification algorithm is run by the verifier. The algorithm's input is public key pk, public state information θ, and proof $P = \{P_d\}$. If the verification succeeds, the output is SUCCESS, and if the verification fails, the result is FALSE.

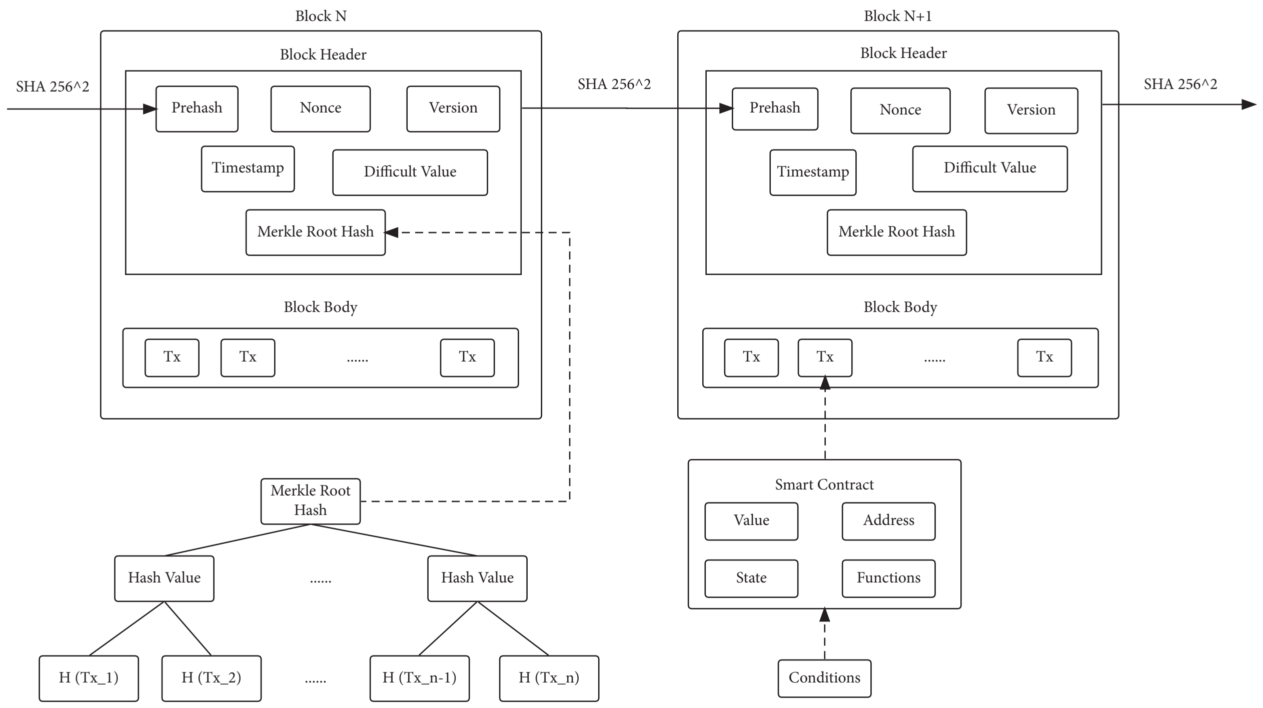

FIGURE 1: Blockchain structure.

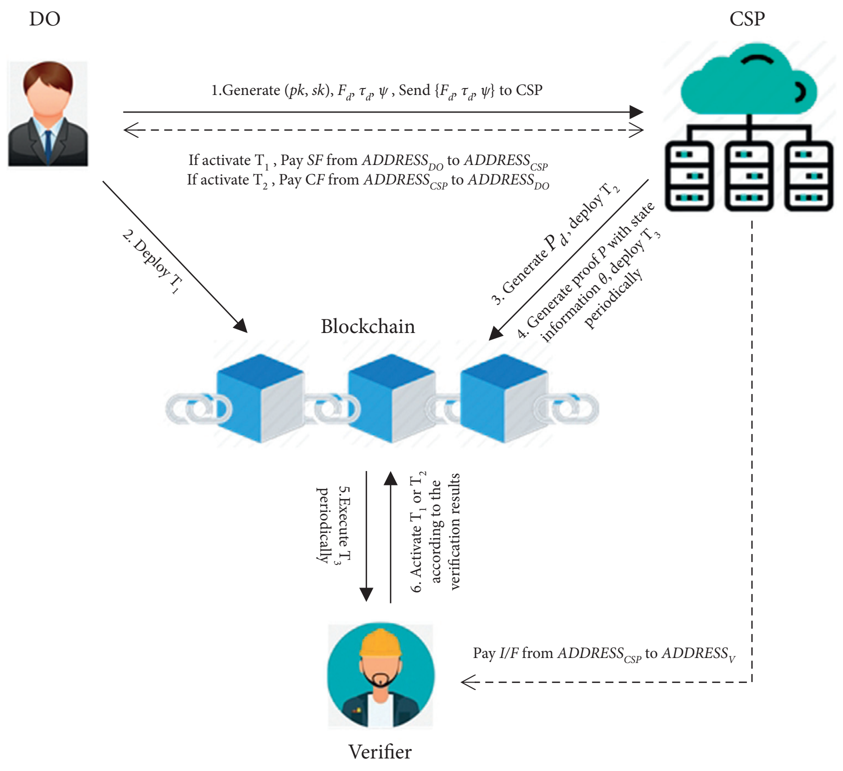
DO
CSP
1.Generate (pk, sk), F_d, τ_d, ψ , Send $\{F_d, \tau_d, \psi\}$ to CSP
If activate T_1 , Pay SF from $ADDRESS_{DO}$ to $ADDRESS_{CSP}$
If activate T_2 , Pay CF from $ADDRESS_{CSP}$ to $ADDRESS_{DO}$
2. Deploy T_1
3. Generate P_d, deploy T_2
4. Generate proof P with state information θ, deploy T_3 periodically
Blockchain
5.Execute T_3 periodically
6. Activate T_1 or T_2 according to the verification results
Pay I/F from $ADDRESS_{CSP}$ to $ADDRESS_V$
Verifier

FIGURE 2: System model.

3.3. Scheme Implementation. In this part, we first introduce the noninteractive multireplica provable data possession (NI-MR-PDP) protocol. Then we propose our noninteractive multireplica data possession scheme combined with smart contract technology.

3.3.1. NI-MR-PDP.

(1). Setup Phase. Let G and G_T be multiplicative cyclic groups of prime order p and g a generator of G, and there is a bilinear map $e\colon G \times G \longrightarrow G_T$. Two hash functions are $H(\cdot)\colon \{0,1\}^* \longrightarrow G$ and $h(\cdot)\colon \{0,1\}^* \longrightarrow Z_p$. Two

pseudorandom functions are$\alpha(\cdot)$: $\{0,1\}^* \longrightarrow \{0,1\}^*$ and $\beta(\cdot)$: $\{0,1\}^* \longrightarrow [1,n]$. Let the number of challenge blocks be an integer c, $1 \leq c \leq n$.

KeyGen: it selects λ as the security parameter, randomly generates a pair of public and secret keys $(spk, ssk) \overset{R}{\leftarrow} SKg$ for signature, and computes $v \leftarrow g^{ssk}$. The public key is $pk = (spk, v)$ and the secret key is $sk = ssk$.

ReplicaGen: DO encrypts the file to get the ciphertext F, divided into n blocks with the same size and expressed as $F = \{f_i\}$, $1 \leq i \leq n$. To prevent the adversary from using files of different replica servers to restore the complete F, DO needs to generate a unique and distinguishable replica file. By adding random values, t various replicas $F_d = \{m_{d,i}\}$ are generated, where $m_{d,i} = f_i + r_{d,i}$, $r_{d,i} = \alpha(d \| i)$, $1 \leq d \leq t$, $1 \leq i \leq n$. The more replicas, the higher reliability of the data, and the corresponding fee charged by CSP will increase.

TagGen: DO randomly selects name$\overset{R}{\leftarrow} Z_p$ and t values $u_1, u_2, \cdots, u_t \overset{R}{\leftarrow} G$ and calculates the tag $\sigma_{d,i} = (H(\text{name}\| i) \cdot u_d^{m_{d,i}})^{sk}$ of each block $m_{d,i}$ in the replica. Due to the aggregation of the BLS signature, the tags of the same subscript blocks of different replicas F_d can be aggregated into $\sigma_i = \prod_{d=1}^{t} \sigma_{d,i}$, and the tag set of blocks is $\psi = \{\sigma_i\}$. Let $\tau_{o\,d} = \text{name}\|n\|d\|u_1\| \cdots \| u_t$, and the tag of each replica is $\tau_d = \tau_{o\,d} \| S\,\text{Sign}_{sk}(\tau_{o\,d})$.

(2). Audit Phase. At this phase, the Verifier does not need to randomly select the challenge set to challenge CSP, like the traditional PDP scheme. Instead, the CSP uses the current public state information θ as the input of the pseudorandom function to simulate the challenge set generation process. In this way, the Verifier can generate a challenge set by itself to meet the requirement of no interaction.

In our scheme, θ should be publicly available and not controlled by the CSP while changing over time. Considering the blockchain structure, the timestamp or hash value of the previous block in the block header can meet the above requirements and be used as θ.

ProofGen: CSP selects an appropriate integer c, $1 \leq c \leq n$. For $\forall j \in [1,c]$ it computes $s_j \leftarrow \beta(\theta \| j)$; we can get $I = \{s_1, s_2, \cdots, s_c\}$. For $\forall i \in I$, it calculates $V_i \leftarrow h(\theta \| j)$, so the challenge set is $Q = \{(i, V_i)\}$, $i \in I$. Each replica server of CSP generates the corresponding proof for its stored replica and computes $\mu_d = \sum_{(i,V_i)\in Q} V_i \cdot m_{d,i}$ and $\sigma = \prod_{(i,V_i)\in Q} \sigma_i^{V_i}$; the proof of replica F_d is $P_d = \{\theta, \tau_d, \mu_d, \sigma\}$, $1 \leq d \leq t$.

ProofVerify: Verifier first compares the state information to verify the correctness of the public state information θ, and if it fails, it returns FALSE, and if it succeeds, it computes $I = \{\beta(\theta \| 1), \beta(\theta \| 2), \cdots, \beta(\theta \| c)\}$ and $V_i \leftarrow h(\theta \| i)$ and then gets the challenge set $Q = \{(i, V_i)\}$, $i \in I$.

Then spk is used to verify the tag of the replica τ_d. If it fails, it returns to FALSE. If it succeeds, it returns $name, n, d$ and $u_1, u_2, \cdots, u_t$.

Finally, we check the equation $e(\sigma, g) \overset{?}{=} e(\prod_{(i,V_i)\in Q} \prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d}, v)$. If the equation holds, output SUCCESS. Otherwise, return FALSE.

It is easy to prove the correctness of the scheme because $v = g^{sk}$, $\sigma_i = \prod_{d=1}^{t} \sigma_{d,i}$, $\sigma_{d,i} = (H(\text{name}\| i) \cdot u_d^{m_{d,i}})^{sk}$, $\mu_d = \sum_{(i,V_i)\in Q} V_i \cdot m_{d,i}$ and $\sigma = \prod_{(i,V_i)\in Q} \sigma_i^{V_i}$; the equation is as follows:

$$\begin{aligned} e(\sigma, g) &= e\left(\prod_{(i,V_i)\in Q} \sigma_i^{V_i}, g\right) \\ &= e\left(\prod_{(i,V_i)\in Q} \left(\prod_{d=1}^{t} \sigma_{d,i}\right)^{V_i}, g\right) \\ &= e\left(\prod_{(i,V_i)\in Q} \left(\prod_{d=1}^{t} \left(H(\text{name}\| i) \cdot u_d^{m_{d,i}}\right)^{sk}\right)^{V_i}, g\right) \\ &= e\left(\prod_{(i,V_i)\in Q} \prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{m_{d,i} \cdot V_i}, g^{sk}\right) \\ &= e\left(\prod_{(i,V_i)\in Q} \prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d}, v\right). \end{aligned} \tag{1}$$

3.3.2. *Smart Contract Scheme.*

In the traditional cloud storage system, DO needs to pay a certain fee to CSP to purchase storage space. Once the DO data is unavailable or tampered with, it is challenging to obtain economic compensation for data rights. On the one hand, DO will no longer store the original data locally, and the proof process will become arduous. On the other hand, the laws and regulations of various countries on data security are not necessarily complete, especially in the case of transnational disputes. Moreover, legal litigation usually means extra money and time costs.

The emergence of smart contracts brings dawn to solve these problems. Since the smart contract has the characteristics of tamper-resistant and automatic triggering, once the contract content is agreed by all parties, as long as the contract conditions are met, the contract results will be implemented immediately. No one can change the contract content again.

Therefore, we design a noninteractive multireplica provable data possession scheme based on smart contracts in the cloud storage system. By deploying smart contracts on the blockchain, we provide tamper-resistant multireplica integrity verification for all parties and guarantee fair payment through the deposit mechanism. A consensus mechanism needs to be used to fight against dishonest verifiers and ensure the correctness of the verifier's smart contract execution. This article does not discuss that in depth.

The flow logic of the scheme is shown in Figure 3, which is described as follows.

(1) DO, CSP, and Verifier register on the blockchain to obtain public-secret key pairs and account addresses ADDRESS_{DO}, ADDRESS_{CSP}, and ADDRESS_V

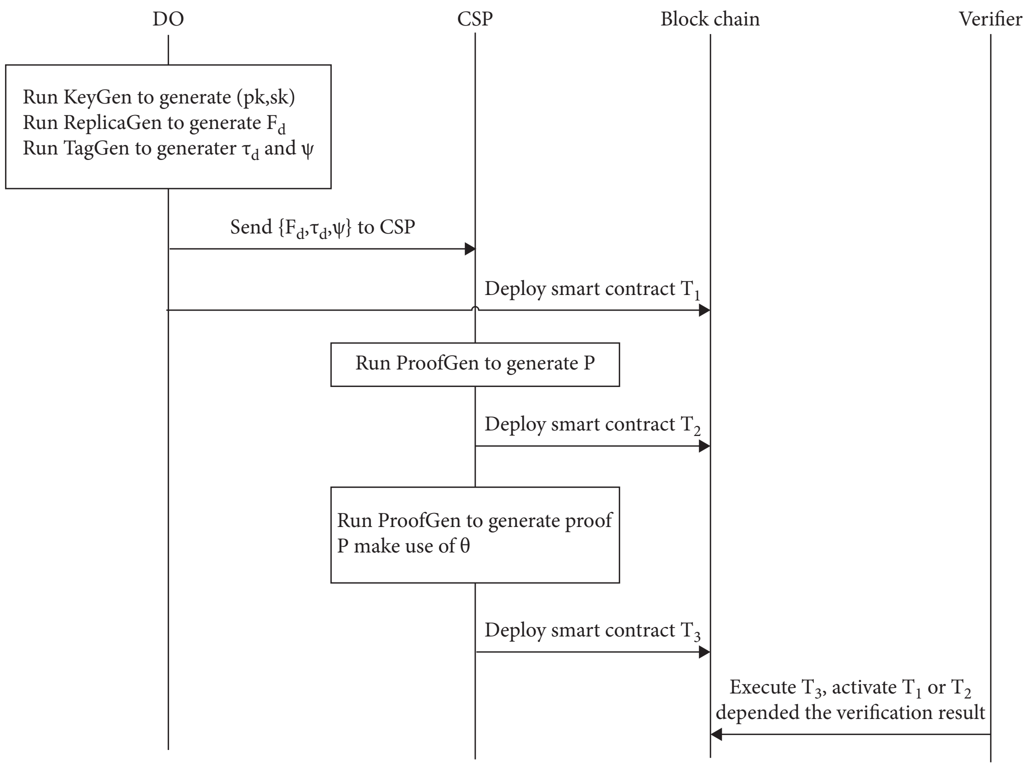
DO
CSP
Block chain
Verifier
Run KeyGen to generate (pk,sk)
Run ReplicaGen to generate F_d
Run TagGen to generater τ_d and ψ
Send {F_d,τ_d,ψ} to CSP
Deploy smart contract T_1
Run ProofGen to generate P
Deploy smart contract T_2
Run ProofGen to generate proof P make use of θ
Deploy smart contract T_3
Execute T_3, activate T_1 or T_2 depended the verification result

Figure 3: The flow logic of the scheme.

respectively. The public-secret key pair is used for signature and verification on the blockchain. The public key usually generates the account address to identify the identity and conduct transactions. DO and CSP need to pay a certain deposit respectively to ensure the smooth completion of subsequent transactions.

(2) DO runs the algorithm KeyGen to generate public-secret key pair (pk, sk), runs the algorithm ReplicaGen to generate t different replicas F_d of file F, and runs the algorithm TagGen to generate the tag of replica τ_d and the tag set of blocks ψ.

(3) DO uploads $\{F_d, \tau_d, \psi\}$ to each replica server of CSP, generates a smart contract T_1 (see Table 1), and deploys it on the blockchain. The smart contract includes the basic information of F (file name, file hash, and upload time), transaction information (storage fee, account address of DO, account address of CSP), and the signature of DO. It can ensure that if the CSP completely stores a replica of the file, it can pass the verification, and DO must pay the cost to the CSP in time.

(4) After receiving $\{F_d, \tau_d, \psi\}$, each replica server of CSP runs the algorithm ProofGen to generate P_d and then sends it to CSP to obtain the proof set $P = \{P_d\}$. CSP generates a smart contract T_2 (see Table 2) and deploys it on the blockchain. The smart contract

Table 1: Storage smart contract T_1.

Smart contract T_1	
File name	FN
File hash	FH
Upload time	UT
Storage fee	SF
Account addresses of DO	ADDRESS_{DO}
Account addresses of CSP	ADDRESS_{CSP}
Signature of DO	Sign_{DO}
Contract content:	
Promise	
{if Proof Verify $(pk, \theta, P) \longrightarrow$ SUCCESS	
Pay SF from ADDRESS_{DO} to ADDRESS_{CSP}}	

Table 2: Compensation smart contract T_2.

Smart contract T_2	
File name	FN
File hash	FH
Receiving time	RT
Compensation fee	CF
Account addresses of DO	ADDRESS_{DO}
Account addresses of CSP	ADDRESS_{CSP}
Signature of DO	Sign_{CSP}
Contract content:	
Promise	
{if Proof Verify $(pk, \theta, P) \longrightarrow$ FALSE	
Pay CF from ADDRESS_{CSP} to ADDRESS_{DO}}	

includes the basic information of F (file name, file hash, and receiving time), transaction information (compensation fee, account address of DO, and account address of CSP), and the signature of CSP. It can ensure that if the CSP does not store complete replicas and the integrity verification fails, the CSP must pay a certain fee to compensate the DO in time.

(5) CSP will periodically generate the corresponding proof combined with the current public state information θ, generate a smart contract T_3 (see Table 3), and deploy it on the blockchain. The smart contract includes the basic information of the file (file name, file hash, generation time of evidence, status information, and evidence information), the contract information to be called, transaction information (verification fee, CSP account address, and verifier account address), and the signature of the CSP. The verifier will execute a smart contract to complete multiple-replica data integrity verification for a reward and then activate T_1 or T_2 based on the verification result.

3.4. Brief Summary. We propose a noninteractive multireplica provable data possession scheme based on smart contract, which not only meets the basic requirements of correctness and security but also has the following characteristics:

(1) Public verification: the evidence verification algorithm is public and does not need to use the private key, so any third party can obtain a public conclusion about whether the data have integrity.

(2) Noninteractive: during the whole verification process, DO and CSP do not need to interact with the third-party verifier, and it DO and CSP do not need to remain online all the time, making the operation of the scheme more flexible.

(3) Batch verification: batch verification can be carried out simultaneously on all replicas. Only one equation needs to be verified, and then it will tell whether all replicas are stored completely.

(4) Fair payment: the payments of DO, CSP, and Verifier follow the agreed smart contract, which cannot be tampered with, and cannot be denied by anyone.

(5) Privacy-preserving: during the whole verification process, the relevant information that the Verifier can access of DO is all encrypted files and cannot obtain any knowledge of DO's original file without knowing the secret key.

4. Security Proof

The security of the scheme is defined by formally describing a security game between challenger C and adversary A:

C generates a public-private key pair (pk, sk) by running KeyGen, sends pk to A, and reminds sk for responding to A's query.

Table 3: Verification smart contract T_3.

Smart contract T_3	
File name	FN
File hash	FH
Proof generated time	PGT
State information	θ
Proof information	P
Storage smart contract	T_1
Compensation smart contract	T_2
Verification fee	VF
Account addresses of CSP	ADDRESS_{CSP}
Account addresses of verifier	ADDRESS_V
Signature of CSP	Sign_{CSP}
Contract content:	
Promise	
{Execute the ProofVerify algorithm if	
Proof Verify$(pk, \theta, P) \longrightarrow$ SUCCESS, activate T_1	
if Proof Verify$(pk, \theta, P) \longrightarrow$ FALSE, activate T_2	
Pay VF from ADDRESS_{CSP} to ADDRESS_V}	

(1) A can query replicas by interacting with C. A randomly selects a file F and sends it to C. C generates t different replicas $F_d\,(1 \leq d \leq t)$ by running the ReplicaGen algorithm and responds to A.

(2) A can query tags by interacting with C. A randomly selected replica F_d and sends it to C, and C generates the tag of replica τ_d and the tag set of blocks ψ by running the TagGen algorithm and responds to A.

(3) A generates proof P' according to responses from multiple queries.

Definition 1. The advantage of adversary A in the game is $A\,dv_A = \Pr[\text{Proof Verify}(pk, \theta, P') = \text{SUCCESS}]$. We say A wins the game if $A\,dv_A$ is nonnegligible.

Definition 2. A noninteractive multireplica provable data possession scheme is secure. If there is an effective extraction algorithm Extr, for any adversary who wins the security game and the output of the proof of file F is P', the probability that the Extr can recover the replicas $\{F_d\}$ (i.e., $Extr(pk, \theta, \tau_d, P') = \{F_d\}$) is nonnegligible.

Theorem 1. *If the signature algorithm used to generate file tags is existential unforgeability, the computational Diffie-Hellman problem on bilinear groups and the discrete logarithm problem are difficult; then, in the random oracle model, the probability that an adversary which breaks the security of our scheme, through the verification algorithm using proof not generated by ProofGen, is negligible.*

We prove the theorem as a series of games with interleaved analysis. The restrictions of the games for the adversary are gradually tightened.

Game-0: Game-0 is the first game, the security game defined at the beginning of this chapter.

Game-1: Game-1 is the same as Game-0, with a slight difference. The challenger keeps a list that stores all signed file tags that have been responded to in the tag query. If the adversary submits an effective tag τ_d but is not in the list signed by the challenger, the challenger outputs failure and aborts.

Analysis: if an adversary causes the challenger outputs failure with nonnegligible probability in Game-1, we can use the adversary to construct a forger to break the unforgeability of the signature scheme.

If the adversary does not cause failure in Game-1, its view is identical in Game-0 and Game-1. Through the description in Game-1, we know that the verification algorithm and extraction algorithm will get the parameters name, n, d and $u_1, u_2, \ldots, u_t$ from the tag τ_d, and these values can only be generated by the challenger.

Therefore, if the adversary's success probability in Game-0 and Game-1 has a nonnegligible difference, we can construct a simulator to break the existence of the signature scheme by using the adversary.

Game-2: Game-2 is the same as Game-1, with a slight difference. The challenger keeps a list of tag queries and responses initiated by all adversaries. If the adversary submits proof that proves the verification algorithm successfully but σ is not equal to $\prod_{(i,V_i)\in Q}\sigma_i^{V_i}$, the challenger outputs failure and aborts.

Analysis: it is assumed that the failed replica file is divided into equal-length n blocks, expressed as $F_d = \{m_{d,i}\}$, $1 \le d \le t$, $1 \le i \le n$, and the corresponding parameters are $name, n, d$ and $u_1, u_2, \cdots, u_t$. The tag set of blocks $\psi$is generated by TagGen. Suppose $Q = \{(i, V_i)\}, i \in I$ is the query that leads to failure, and the proof that responds to the adversary is $\mu_t', \mu_t', \ldots, \mu_t'$ and σ'. Let the expected response generated by an honest prover be $\mu_1, \mu_2, \ldots, \mu_t$ and σ, where $\mu_d = \sum_{(i,V_i)\in Q} V_i \cdot m_{d,i}$ and $\prod_{(i,V_i)\in Q}\sigma_i^{V_i}$. According to the proof of correctness, we know that the expected response satisfies the equation $e(\sigma, g) = e(\prod_{(i,V_i)\in Q}\prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d}, v)$. According to the description of Game-2, the adversary's response can also satisfy the equation $e(\sigma', g) = e(\prod_{(i,V_i)\in Q}\prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d'}, v)$, but $\sigma' \ne \sigma$. If there is $\mu_d' = \mu_d$ for each d, it satisfies the equation $\sigma' = \sigma$, which contradicts the above assumption. Therefore, let $\Delta\mu_d = \mu_d' - \mu_d$, and we know that at least one of $\{\triangle\mu_d\}$ is not 0.

Now we prove that if the adversary leads to the challenger outputs failure in Game-2 with nonnegligible probability, we can construct a simulator to solve the computational Diffie-Hellman problem.

The input value of the simulator is g, g^{sk}, $h \in G$, and its goal is to output h^{sk}. The behavior of the simulator is similar to the challenger in Game-1, but there are the following differences:

(1) When generating the key, it sets the public key to g^{sk} received in the challenge, which means the simulator does not know the secret key sk.

(2) The simulator programs the random oracle H and keeps a list of queries and responses. When the adversary randomly selects $r \overset{R}{\leftarrow} Z_p$ to query, its response is $g^r \in G$. It also responds to queries $H(\text{name}\| i)$ in a particular way, seen later.

(3) When asked to store some file whose coded representation comprises the n blocks $\{m_{d,i}\}$, $1 \le d \le t$, $1 \le i \le n$, the simulator behaves as follows. It chooses a name name$\overset{R}{\leftarrow} Z_p$ at random. Because the space for choosing the name is large enough, the probability that the simulator chooses a name that has been queried by *name* $\|$ i in random oracle H is negligible.

For each $d, 1 \le d \le t$, the simulator chooses random values $\delta_d, \gamma_d \overset{R}{\leftarrow} Z_p$ and set $u_d = g^{\delta_d} \cdot h^{\gamma_d}$. For each i, $1 \le i \le n$, the simulator chooses a random value $r_i \overset{R}{\leftarrow} Z_p$, and the response of the random oracle H is

$$H(\text{name}\| i) = \frac{g^{r_i}}{\left(g^{\sum_{d=1}^{t}\delta_d \cdot m_{d,i}} \cdot h^{\sum_{d=1}^{t}\gamma_d \cdot m_{d,i}}\right)}. \quad (2)$$

Now the simulator can calculate because we have

$$\begin{aligned} &H(\text{name}\| i) \cdot \prod_{d=1}^{t} u_d^{m_{d,i}} \\ &= \left(\frac{g^{r_i}}{\left(g^{\sum_{d=1}^{t}\delta_d \cdot m_{d,i}} \cdot h^{\sum_{d=1}^{t}\gamma_d \cdot m_{d,i}}\right)}\right) \cdot \prod_{d=1}^{t} u_d^{m_{d,i}} \\ &= \left(\frac{g^{r_i}}{\left(g^{\sum_{d=1}^{t}\delta_d \cdot m_{d,i}} \cdot h^{\sum_{d=1}^{t}\gamma_d \cdot m_{d,i}}\right)}\right) \\ &\cdot \left(g^{\sum_{d=1}^{t}\delta_d \cdot m_{d,i}} \cdot h \sum_{d=1}^{t}\gamma_d \cdot m_{d,i}\right) \\ &= g^{r_i}. \end{aligned} \quad (3)$$

Therefore

$$\begin{aligned} \sigma_i &= \prod_{d=1}^{t}\sigma_{d,i} = \prod_{d=1}^{t}\left(H(\text{name}\| i) \cdot u_d^{m_{d,i}}\right)^{sk} \\ &= \left(H(\text{name}\| i) \cdot \prod_{d=1}^{t} u_d^{m_{d,i}}\right)^{sk} \\ &= \left(g^{sk}\right)^{r_i}. \end{aligned} \quad (4)$$

(4) The simulator continues to interact with the adversary until the particular situation defined by Game-2 occurs: the adversary successfully proves the verification with σ' that is different from the expected σ.

The analysis of Game-0 and Game-1 ensures that the parameters $name, n, \{u_d\}, \{m_{d,i}\}, \{\sigma_i\}$ used in the protocol are generated by the challenger; otherwise, it will output failure. This means that these parameters are generated gradually by the simulator described above. By dividing the tag σ' and the expected tag σ, we get

$$\begin{aligned} e\left(\frac{\sigma'}{\sigma}, g\right) &= e\left(\prod_{d=1}^{t} u_d^{\Delta\mu_d}, v\right) \\ &= e\left(\prod_{d=1}^{t} \left(g^{\delta_d} \cdot h^{\gamma_d}\right)^{\Delta\mu_d}, v\right) \\ &= e\left(g^{\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d} \cdot h^{\sum_{d=1}^{t}\gamma_d \cdot \Delta\mu_d}, v\right). \end{aligned} \quad (5)$$

Rearranging terms yields

$$e\left(\sigma' \cdot \sigma^{-1} \cdot v^{-\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d}, g\right) = e\left(h^{\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d}, v\right). \quad (6)$$

Because $v = g^{sk}$, we find that the computational Diffie-Hellman problem has been solved:

$$h^{sk} = \left(\sigma' \cdot \sigma^{-1} \cdot v^{-\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d}\right)^{-\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d} \quad (7)$$

Unless the denominator is 0, we know that at least one of $\{\triangle\mu_d\}$ is not 0, so the probability of the denominator being 0 can be ignored.

Therefore, we prove that if there is a nonnegligible difference between the adversary's probability of success in Game-1 and Game-2, we can construct a simulator to solve the computational Diffie-Hellman problem, as required.

Game-3: Game-3 is the same as Game-2, with a slight difference. If the adversary submits proof that explains the verification successfully, but at least one μ_d is not equal to $\sum_{(i,V_i)\in Q} V_i \cdot m_{d,i}$, the challenger outputs failure and aborts.

Analysis: make some definitions like Game-2. We suppose that the failed replica file is divided into equal-length n blocks, expressed as $F_d = \{m_{d,i}\}$, $1 \le d \le t$, $1 \le i \le n$, and the corresponding parameters are $name, n, d$ and $u_1, u_2, \cdots, u_t$. The tag set of blocks ψ is generated by TagGen. Suppose $Q = \{(i, V_i)\}, i \in I$ is the query that leads to failure, and the proof that responds to the adversary is $\mu_1', \mu_2', \ldots, \mu_t'$ and σ'. Let the expected response generated by an honest prover be $\mu_1, \mu_2, \cdots, \mu_t$ and σ, where $\mu_d = \sum_{(i,V_i)\in Q} V_i \cdot m_{d,i}$ and $\sigma = \prod_{(i,V_i)\in Q} \sigma_i^{V_i}$. Game-2 has ensured that we get $\sigma' = \sigma$; only $\{\mu_d'\}$ and $\{\mu_d\}$ can be different. Define $\Delta\mu_d = \mu_d' - \mu_d$, $1 \le d \le t$; then at least one of $\{\Delta\mu_d\}$ is not 0.

Now we prove that if the adversary leads to the challenger outputs failure in Game-3 with nonnegligible probability, we can build a simulator to solve the discrete logarithm problem.

The input value of the simulator is $g, h \in G$, and its goal is to output x such that $h = g^x$. The behavior of the simulator is similar to the challenger in Game-2, but there are the following differences.

(1) When it is required to store a file whose coded representation comprises the n blocks $\{m_{d,i}\}$, $1 \le d \le t$, $1 \le i \le n$, the simulator behaves according to TagGen. For each d, $1 \le d \le t$, the simulator chooses random values $\delta_d, \gamma_d \overset{R}{\leftarrow} Z_p$ and sets $u_d = g^{\delta_d} \cdot h^{\gamma_d}$.

(2) The simulator continues to interact with the adversary until the particular situation defined by Game-3 occurs: the adversary successfully proves the verification with $\{\mu_d'\}$ different from the expected $\{\mu_d\}$.

According to the analysis of Game-1, we know that the parameters name, n, $\{u_d\}$, $\{m_{d,i}\}$, $\{\sigma_i\}$ used in the protocol are generated by the simulator. According to the analysis of Game-2, we know $\sigma' = \sigma$.

Construct the verification equation with $\{\mu_d\}$, respectively, with

$$\begin{aligned} &e\left(\prod_{(i,V_i)\in Q}\prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d}, v\right) \\ &= e(\sigma, g) \\ &= e(\sigma', g) \\ &= e\left(\prod_{(i,V_i)\in Q}\prod_{d=1}^{t} H(\text{name}\| i)^{V_i} \cdot u_d^{\mu_d}, v\right), \end{aligned} \quad (8)$$

concluding that

$$\prod_{d=1}^{t} \mathrm{u}_d^{\mu_d} = \prod_{d=1}^{t} \mathrm{u}_d^{\mu_d'}, \quad (9)$$

and therefore

$$\begin{aligned} 1 &= \prod_{d=1}^{t} \mathrm{u}_d^{\Delta\mu_d} \\ &= \prod_{d=1}^{t} \left(g^{\delta_d} \cdot h^{\gamma_d}\right)^{\Delta\mu_d} \\ &= g^{\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d} \cdot h^{\sum_{d=1}^{t}\gamma_d \cdot \Delta\mu_d}. \end{aligned} \quad (10)$$

We find that the discrete logarithm problem has been solved:

$$h^{sk} = \left(\sigma' \cdot \sigma^{-1} \cdot v^{\sum_{d=1}^{t}\delta_d \cdot \Delta\mu_d}\right)^{\sum_{d=1}^{t}\gamma_d \cdot \Delta\mu_d}. \quad (11)$$

Unless the denominator is 0. However, we know that at least one of $\{\Delta\mu_d\}$ is not 0, so the probability of the denominator being 0 can be ignored.

Therefore, we prove that if there is a nonnegligible difference between the adversary's probability of success in Game-2 and Game-3, we can construct a simulator to solve the discrete logarithm problem, as required.

4.1. Wrapping Up. Suppose the signature algorithm used to generate file tags is existential unforgeability, the computational Diffie-Hellman problem on bilinear groups and the discrete logarithm problem are difficult. In that case, there is a nonnegligible difference between the adversary's probability of success in Game-3 and Game-0. From Game-1 to

Table 4: Comparison of schemes' characteristics.

	[17]	[18]	[19]	[20]	[21]	[22]	Our scheme
Public verification	√	√	√	√	√	√	√
Batch verification	×	√	√	×	×	√	√
Privacy preserving	×	×	√	√	√	√	√
Fair payment	√	√	√	√	√	√	√
Interactivity	√	×	×	√	×	×	√
Multireplica	×	√	×	×	√	√	√

Table 5: Algorithm running time of our scheme ($N = 64$ KB, $n = 1024$, $c = 300$, $t = 3$).

Algorithm	ReplicaGen	TagGen	ProofGen	ProofVerify	Total
Time(s)	0.009413	3.376702	0.222226	0.003383	3.611731

Game-3, it is limited that τ_d, σ, μ_d, only the proof not correctly calculated by ProofGen, can respond to the challenge gradually, and Game-0 is the security game of our scheme. Therefore, the probability that an adversary who breaks through the security of our scheme uses the proof not generated by ProofGen to successfully prove the verification is negligible. This completes the proof of Theorem 1.

5. Performance Evaluation

5.1. Comparative Analysis. We compare our scheme with other similar schemes from the dimensions of public verification, batch verification, privacy preserving, fair payment, interactivity, and multireplica. It can be seen from Table 4 that compared with other blockchain-based provable data possession schemes. Our scheme uses multireplica technology, based on BLS signature and homomorphic verification tag, combined with the smart contract with deposit mechanism, and it achieves all functions well.

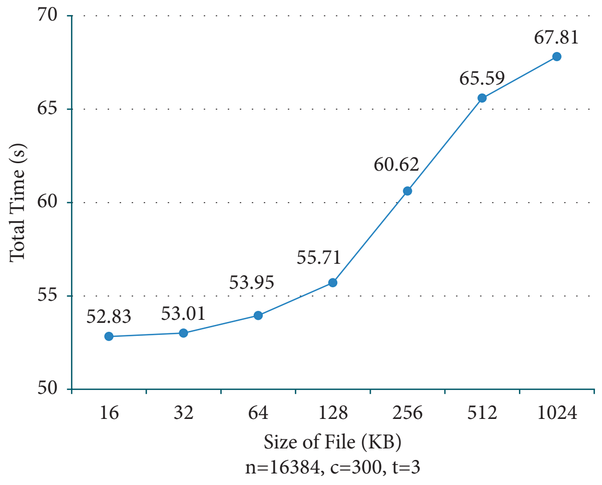

Figure 4: The relationship between total time and file size ($n = 16384$, $c = 300$, $t = 3$).

5.2. Experiments. To evaluate the performance of our scheme, we designed a prototype of the scheme with C Programming language. For the large integer and pairing operations, we use the GMP Library (version 6.2.1) and PBC Library (version 0.5.14), respectively. AES256 is used for file encryption, and the length of signature key is 160 bits. The experimental environment is Intel Core i7 2.6 GHz, memory is 16 GB 2133mhz lpddr3, and the operating system is macOS 12.0.1.

In the experiment, we mainly focus on the total running time of the scheme, including ReplicaGen time, TagGen time, ProofGen time, and ProofVerify time, as well as the relationship between the running time and file size N, number of blocks n, and number of challenge blocks c.

According to the study of [2], it assumes that the CSP destroys 1% of the data blocks uploaded by the DO; when $c = 300$ and 460, the probability that DO can detect misbehavior of CSP is 95% and 99%.

The fixed file size is $N = 64$ KB, the number of blocks is $n = 1024$, the number of challenge blocks is $c = 300$, and the number of replicas is $t = 3$. The experimental results of the running time of each algorithm are shown in Table 5. It can be seen from Table 5 that time is mainly consumed in the TagGen algorithm, which is consistent with our assumption. In this algorithm, tags need to be calculated for each data block, and many exponential operations need to be carried out. Hence, the computational complexity is much higher than that in other algorithms.

Next, we fixed the number of blocks $n = 16384$, the number of challenge blocks $c = 300$, and the number of replicas $t = 3$ and gradually increased the file size N from 16 KB to 1024 KB. The experimental results are shown in Figure 4. We can see that the relationship between the total time and the file size is basically linear, and the gap is not apparent when N is very small.

Then we fixed the file size $n = 64$ KB, the number of challenge blocks $C = 300$, gradually changed the number of blocks from $n = 1024$ to 16384, and took the number of replicas t as 1, 2, and 3, respectively. The experimental results are shown in Figure 5. We can see that the total time increases exponentially with the number of blocks, and the total time increases with the number of replicas, but the impact is limited.

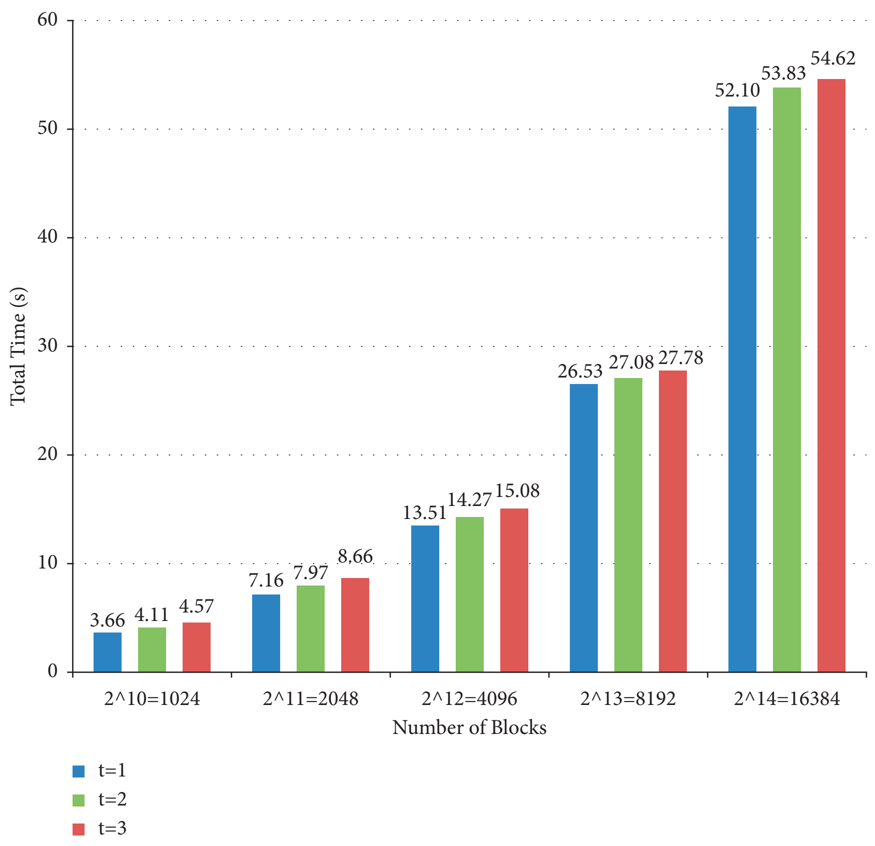

FIGURE 5: The relationship between total time and the number of blocks and replicas ($N = 64$ KB, $c = 300$).

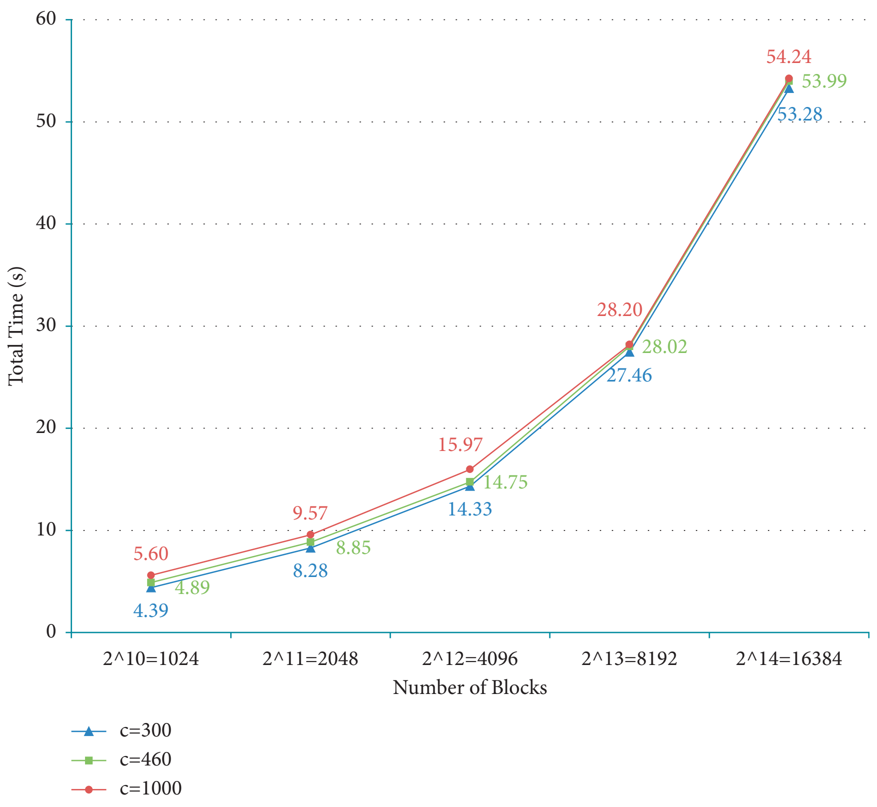

FIGURE 6: The relationship between total time and the number of blocks and challenge blocks ($N = 64$ KB, $t = 3$).

Finally, we fixed the file size $N = 64$ KB and the number of replicas $t = 3$, gradually changed the number of blocks from $n = 1024$ to 16384, and took the number of challenge blocks c as 300, 460, and 1000, respectively. The experimental results are shown in Figure 6. We can see that the total time increases with the number of challenge blocks, but the impact of the number of challenge blocks on the total time becomes smaller and smaller as the number of blocks increases.

6. Conclusion

This paper proposes a noninteractive multireplica provable data possession scheme based on smart contracts. The scheme replaces TPA with the smart contract, which provides a trusted environment for data integrity verification so that the verification process is public, tamper-proof, traceable, and periodically automatic. To reduce the transaction fees caused by the frequent operation of blockchain in the verification process, the concept of noninteractive is introduced. By presetting payment rules in smart contracts, fair transactions between the parties involved are guaranteed. The contracts will be executed automatically once the conditions are met.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (nos. 2020YFB1708600 and 2020YFB1805403) and Foundation of Guizhou Provincial Key Laboratory of Public Big Data (nos. 2017BDKFJJ015, 2018BDKFJJ008, 2018BDKFJJ020, and 2018BDKFJJ021).

References

[1] Seagate Rethink Data, *Put More of Your Business Data to Work—From Edge to Cloud*, Seagate Rethink Data, Chennai, India, 2020.

[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*, ACM, NY, USA, October 2007.

[3] A. Juels and J. B. Kaliski, "Pors: Proofs of Retrievability for Large Files," in *Proceedings of the 14th ACM conference on Computer and communications security*, ACM, NY, USA, October 2007.

[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90–107, Melbourne, VIC, Australia, December 2008.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," *Computer Security-ESORICS 2009*, vol. 5789, pp. 355–370, 2009.

[6] C. Wang, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE INFOCOM*, San Diego, CA, USA, March 2010.

[7] R. Curtmola, "Multiple-Replica Provable Data Possession," in *Proceedings of the 2008 the 28th International Conference on Distributed Computing Systems*, IEEE, Beijing, China, June 2008.

[8] Z. Hao and N. Yu, "A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability," in *Proceedings of the 2010 Second International Symposium on Data, Privacy, and E-Commerce*, IEEE, Buffalo, NY, USA, September 2010.

[9] J. Wei, "Efficient dynamic replicated data possession checking in distributed cloud storage systems," *International Journal of Distributed Sensor Networks*, vol. 2016, Article ID 1894713, 2016.

[10] Z. Ya-xing, "Multiuser and multiple-replica provable data possession scheme based on multi-branch authentication tree," *Journal on Communications*, vol. 36, no. 11, pp. 80–91, 2015.

[11] S. Peng, F. Zhou, J. Li, Q. Wang, and Z. Xu, "Efficient, dynamic and identity-based remote data integrity checking for multiple replicas," *Journal of Network and Computer Applications*, vol. 134, pp. 72–88, 2019.

[12] H. Yu, Z. Yang, M. Waqas et al., "Efficient dynamic multireplica auditing for the cloud with geographic location," *Future Generation Computer Systems*, vol. 125, pp. 285–298, 2021.

[13] Y. Chen, J. Sun, Y. Yang, T. Li, X. Niu, and H. Zhou, "PSSPR: a source location privacy protection scheme based on sector phantom routing in WSNs," *International Journal of Intelligent Systems*, vol. 37, no. 2, pp. 1204–1221, 2022.

[14] T. Li, L. Chunmei, J. Yanling, and Y. Yixian, "Is semi-selfish mining available without being detected?" *International Journal of Intelligent Systems*, pp. 1–22, 2021.

[15] T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, and X. Yu, "Semi-selfish mining based on hidden Markov decision process," *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3596–3612, 2021.

[16] Y. Chen, S. Dong, T. Li, Y. Wang, and H. Zhou, "Dynamic multi-key FHE in asymmetric key setting from LWE," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5239–5249, 2021.

[17] T. Li, "Rational Protocols and Attacks in Blockchain System," *Security and communication networks*, vol. 2020, Article ID 8839047, 2020.

[18] J. Xue, C. Xu, and L. Bai, "DStore: a distributed system for outsourced data storage and retrieval," *Future Generation Computer Systems*, vol. 99, pp. 106–114, 2019.

[19] J. Xue, "Identity-based Public Auditing for Cloud Storage Systems against Malicious Auditors via Blockchain," *Science China-Information Sciences*, vol. 62, Article ID 0321043, 2019.

[20] X. Yang, X. Pei, M. Wang, T. Li, and C. Wang, "Multireplica and multi-cloud data public audit scheme based on blockchain," *IEEE ACCESS*, vol. 8, pp. 144809–144822, 2020.

[21] P. Huang, K. Fan, H. Yang, K. Zhang, H. Li, and Y. Yang, "A collaborative auditing blockchain for trustworthy data integrity in cloud storage system," *IEEE ACCESS*, vol. 8, pp. 94780–94794, 2020.

[22] Y. Xu, “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.

[23] 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.

[24] K. Fan, Z. Bao, M. Liu, A. V. Vasilakos, and W. Shi, “Dredas: decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT,” *Future Generation Computer Systems*, vol. 110, pp. 665–674, 2020.

[25] 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.

[26] 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.

[27] R. Chen, Y. Li, Y. Yu, H. Li, X. Chen, and W. Susilo, “Blockchain-based dynamic provable data possession for smart cities,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4143–4154, 2020.

[28] B. L. H. S. Dan Boneh, “Short Signatures from the Weil Pairing,” in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Gold Coast, QLD, Australia, December 2001.

[29] “Bitcoin: A peer-to-peer electronic cash system,” 2009, https://bitcoin.org/en/bitcoin-paper.

[30] N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday*, vol. 2, no. 9, 1997.

[31] P. McCorry, S. F. Shahandashti, and F. Hao, “A smart contract for boardroom voting with maximum voter privacy,” *Financial Cryptography and Data Security*, Springer International Publishing, vol. 10322, pp. 357–375, Cham, 2017.

[32] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, “Blockchain and smart contracts for insurance: is the technology mature enough?” *Future Internet*, vol. 10, no. 2, p. 20, 2018.