Efficient Integrity Auditing Mechanism With Secure Deduplication for Blockchain Storage

Qingyang Zhang, Dongfang Sui, Jie Cui, *Senior Member, IEEE*, Chengjie Gu, and Hong Zhong, *Member, IEEE*

***Abstract*—Massive nodes in a blockchain form an off-chain distributed storage network to provide storage resources for users to meet large data upload requirements. However, this storage approach introduces security and performance issues. Firstly, it is difficult to guarantee the integrity of the data uploaded, and these data may be easily corrupted or lost. Moreover, uploading excessive duplicate data leads to a waste of storage resources. In this study, to address these issues, with a double-copy storage model for blockchain off-chain storage, a novel public auditing scheme with client-side deduplication is proposed to reduce the storage overhead of nodes and check the integrity of the off-chain data. Based on smart contracts, our scheme could realize efficient user ownership and off-chain data integrity verification automatically. In addition, both data encryption and deduplication are achieved based on message-locked encryption and an improved authenticator generation algorithm. Security analysis and experimental comparisons show that the proposed scheme is effective and practical.**

***Index Terms*—Blockchain, client-side deduplication, off-chain distributed storage, privacy-preserving, public auditing, smart contract.**

I. Introduction

In RECENT years, blockchain [1] has become increasingly significant with the development of society and technology. Many applications have been proposed built on top of the blockchain, such as medical information sharing and supply chain tracking. All of these applications will store their data in the blockchain. In addition, owing to the characteristics of decentralization and anonymity of blockchain, many users tend to upload some sensitive data to the blockchain, ensuring that the data are unforgeable and tamper-proof. However, according to some studies [2], [3], the slow packing speed and small capacity of current blocks make satisfying the growing demand of users difficult for large data uploads. To solve the scalability problem in blockchain, the off-chain storage model is introduced [4], where an off-chain storage network is formed by blockchain nodes or other storage services, and metadata is stored in an on-chain ledger. Thus, the storage space and efficiency are improved. However, there still exists a big security issue for blockchain off-chain storage.

In the traditional blockchain, the data is stored in the ledger, which is distributed with copies in all blockchain nodes. The failure of one node will not affect the data reliability. However, for the off-chain model, the data might only be stored in one node without any other copy. After uploading the data to the blockchain, the user might delete the source data. In this case, the data will be lost if a malicious off-chain node deletes that data marked as cold data. Usually, data auditing schemes allow users to verify data integrity remotely by checking partial data blocks with random challenge values without downloading the entire data. Ateniese et al. [5] proposed the concept of provable data possession (PDP) to achieve efficient remote data auditing. Moreover, improved schemes [6], [7], [8] have been proposed to promote the development of data auditing. Thus, to ensure data integrity and improve the availability of storage services, an auditing mechanism that protects the data in off-chain storage nodes is necessary.

Unfortunately, only data auditing is not enough. Based on [9], approximately 75% of the data stored in a storage provider is repeatedly uploaded by users. Thus, data deduplication is also should be considered in data auditing for blockchain off-chain storage, which could significantly reduce storage burden. Especially encrypted data deduplication is more pratical for blockchain storage, which could protect the data accessed by off-chain nodes. The user usually prefers to store the data in the ciphertext. Therefore, to save storage resources, convergent encryption (CE) [10] has been proposed to implement encrypted data deduplication [11], where different users can encrypt the same plaintext into the same ciphertext such that the storage providers can detect and remove duplicate copies of encrypted data. Based on the benefits afforded by CE, many improved schemes [12], [13], [14], [15] have been proposed to enhance deduplication. Also, some data auditing schemes [16], [17] with

Manuscript received 5 September 2022; revised 5 January 2023; accepted 11 February 2023. Date of publication 23 February 2023; date of current version 11 July 2023. The work was supported in part by the National Natural Science Foundation of China under Grants 62272002, 62202005, and U1936220, in part by the Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31, in part by the Natural Science Foundation of Anhui Province, China under Grant 2208085QF198, in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049, and in part by the Special Fund for Key Program of Science and Technology of Anhui Province, China under Grant 202003A05020043. Recommended for acceptance by D. He. *(Corresponding author: Hong Zhong.)*

Qingyang Zhang, Dongfang Sui, Jie Cui, and Hong Zhong are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China, and also with the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: qingyang.zhang.inchina@gmail.com; sdf_1998sdf@163.com; cuijie@mail.ustc.edu.cn; zhongh@ahu.edu.cn).

Chengjie Gu is with the School of Public Security and Emergency Management, Anhui University of Science and Technology, Hefei 231131, China, and also with the Security Research Institute, New H3C Group, Hefei 230088, China (e-mail: gcj@ustc.edu.cn).

Digital Object Identifier 10.1109/TC.2023.3248278

data deduplication support are proposed by combined with the above CE schemes.

However, these auditing schemes cannot be directly used for blockchain, especially for the off-chain storage model [4]. Many existing schemes make a third-party auditor (TPA) to generate random challenge values and perform the auditing task. However, it is difficult in blockchain where no one centralized node to generate such values. In this case, some blockchain-based schemes use blockchain nonce value to generate challenge values [18], [19]. However, current blockchain-based auditing schemes are designed to perform data auditing for a storage service, or multiple storage services, which store multiple copies of data. These cannot be used to audit the off-chain data with a fully distributed storage networks. Moreover, many data auditing schemes require using both files and authenticators to verify data integrity. However, these schemes tend to focus only on file deduplication without addressing authenticator deduplication. According to [20], authenticators may occupy more storage space than the data blocks. Therefore, achieving authenticator deduplication is critical for more effective space-savings.

For these challenges and issues in auditing schemes, a practical and secure data auditing scheme with deduplication support for distributed off-chain storage model remain lacking.

A. Contributions

According to the preceding discussion, while off-chain distributed storage is a desirable option, existing schemes do not achieve both encrypted data and authenticator elimination, or even public ownership and integrity verification. Therefore, based on the existing research work, in this study, an efficient integrity auditing mechanism with secure deduplication for blockchain off-chain storage is proposed. The main contributions are as follows.

- Focusing on data availability and duplication issues, our scheme uses a double-copy storage model for off-chain distributed storage to protect user data from a single point of failure, while using the HCE2 algorithm and improved authenticator generation algorithm to achieve both data and authenticator deduplication, as well as ensure consistency of user data. In addition, our scheme ensures that users who upload duplicate data only provide proof of ownership to the smart contract instead of encrypting all data, thereby considerably reducing computational overhead and metadata redundancy.
- Considering the truthfulness of data integrity audit results, the proposed scheme uses smart contracts as a trusted audit platform to achieve public auditing. Based on this, a novel auditing protocol is designed to verify the integrity of data stored in off-chain storage nodes, and the audit results are published on the blockchain to avoid repeated audit tasks.
- Our scheme supports batch auditing to improve the efficiency of data auditing. Theoretical analysis and experiments performance show that our scheme has lower overhead and can effectively protect user data.

B. Organization of the Rest Paper

The rest of this study is organized as follows. Related research on data auditing and deduplication is reviewed in Section II. Section III presents the preliminary content and definitions. The detail of the scheme is presented in Section IV. The security of this scheme is analyzed in Section V. Section VI presents and analyzes the experimental results. Finally, we conclude this study and describe the future work in Section VII.

II. Related Work

To verify the integrity of users' data in storage servers, various data auditing schemes have been proposed. The notion of PDP was introduced by Ateniese et al. [5], which uses a random sampling strategy to effectively verify data integrity without downloading the complete data. Since then, several PDP-based auditing schemes have been proposed to enhance security and efficiency. However, these schemes are not friendly for users with low computational resources. To reduce the computing burden on users, a dynamic PDP mechanism was proposed by Zhao et al. [21], which signed a contract with a third-party auditor (TPA) to achieve data auditing and can reduce computation overhead of TPA. To protect users' privacy, Yang et al. [22] proposed an identity-based PDP mechanism that achieves encrypted data integrity verification. Considering the retrievability of users' data, a novel notion called Proof-of-retrievability (PoR) was proposed by Juels et al. [6] based on the sentinel technique. To improve the audit efficiency, Shacham and Waters [7] proposed an improved PoR mechanism by introducing erasure coding and BLS aggregate signature [23], which supports batch auditing.

However, TPAs as untrustworthy third parties may face audit trust issues. The rise of blockchain [1], [24] has driven scheme improvements. Zhang et al. [25] proposed a certificateless public verification scheme, where TPA is required to record each verification result on the blockchain. Moreover, if the TPA faces a single point of failure, subsequent audit tasks will not be carried out. To solve the issue, Wang et al. [26] designed a non-interactive data auditing scheme, where the challenge process no longer requires a TPA, but instead leverages blockchain public information. Li et al. [27] presented an MHT-based [28] auditing scheme where the hash tags are stored on the blockchain and used to regenerate an MHT. Although these schemes use blockchain to store public verification information, the verification process is performed by users, which increases user communication and computational overhead. The development of blockchain smart contracts brings new approaches to trusted auditing. Xu et al. [29] proposed a contract-based auditing scheme, where smart contracts act as trusted auditors to ensure the authenticity of audit results and resist single points of failure. Furthermore, Yuan et al. [30] used smart contracts to execute fair arbitration to protect the interests of users whose data is corrupted by a malicious cloud storage provider. However, the above scheme consumes a large amount of gas for smart contract computation, which is impractical in a real environment. In some studies, to address the latency problem that exists in the centralized audit model of edge computing [31], Li et al. [32] designed distributed consensus mechanisms where edge devices can collaborate with each other to verify data integrity. But without proper incentives, edge servers may not participate in consensus. To address this challenge, blockchain-based EdgeWatch [33] was proposed to incentivize edge devices.

Moreover, eliminating redundant data based on integrity auditing to reduce the storage burden on the storage provider is still a problem. None of the above schemes consider data deduplication. Depending on whether the stored data is encrypted or not, deduplication techniques [34] are divided into two categories: deduplication of plaintext and deduplication of ciphertext. It is easy to achieve plaintext deduplication. However, using conventional encryption algorithms (e.g., AES, DES) to encrypt files is difficult to remove duplicate files. Because different users encrypt same files into different ciphertexts using different private keys. For privacy-preserving deduplication, the thought of CE was presented by Douceur et al. [10] used to remove the duplicate encrypted data. Subsequently, to lessen the risk of data being illegally obtained by malicious users due to file tags leakage in CE, the notion of message-locked encryption (MLE) was introduced by Bellare et al. [12] To achieve efficient deduplication and save more storage space, some schemes [35], [36] implemented block-level deduplication. However, too many encryption operations and ciphertext deduplication also consume a lot of computational resources. To overcome this challenge, Miao et al. [37] first proposed a protocol using proofs-of-ownership (PoWs) based on the chameleon hash function without key exposure, which enables users to efficiently convince a verifier that he/she indeed owns entire data.

To maximize the benefits of integrity auditing, Zheng and Xu [38] introduced a concept called proof of storage with deduplication (POSD), which aims to protect the integrity of user data while achieving deduplication. However, this model is still deficient. The combination of the two modules causes a huge computational and storage overhead originating from overgenerating file tags and authenticators. To solve the these issues, Xu et al. [19] devised an elaborate scheme, where authenticators used in integrity verification are applied to proof of ownership. To protect data privacy and eliminate untrustworthy TPA, Tian et al. [39] proposed a proof of storage scheme with encrypted data deduplication based on blockchain, which protects data privacy while achieving shared auditing by using double-server model.

The single point of failure issue of centralized storage is also worth being considered compared to TPA. To improve data availability, massive auditing schemes based on decentralized storage have been proposed. Storj [40] stored files decentralized on different personal computers and introduced a private storage auditing framework, whose storage proofs are publicly available and verifiable. However, it still has some centralization problems. Siacoin [41] is a decentralized cloud storage platform based on blockchain. It sends encrypted file chunks to distributed network storage, and users manage the data through private keys. Filecoin [42], based on IPFS, used a unique proof of space and time (PoST) consensus mechanism to verify that each node is actually saving data. However, the major deficiency of filecoin is that there is no data persistence guarantee mechanism. The above three distributed storage schemes provide more decentralized storage resources for more users and increase the availability of data. Besides, Tian et al. [39] presented a double-server POSD protocol that executes ciphertext PoWs and achieves shared auditing. However, this audit approach may threaten server collusion and affect the truthfulness of integrity verification results. Thus, it is important to achieve data auditing as well as client-side file and authenticator deduplication in off-chain distributed storage in a trustworthy manner.

III. Preliminaries and Definitions

A. Preliminaries

1) Hash-and-Convergent-Encryption-2 (HCE2): It is derived from Message-Locked Encryption (MLE) [12]. Different from encrypting data using their own private key, users first get the convergence keys by calculating the hash of the data blocks and then encrypt the original data blocks using calculated keys. Based on this, different users who have the same data can obtain the same key and ciphertext. Thus, it succeeds in achieving ciphertext deduplication. In addition, for the issue that the consistency of data downloaded by users from the server may be corrupted, HCE2 designs a tag-checking mechanism. The user will regenerate the labels using plaintext after decryption and compare them with the original labels. Users accept the data only if the tags are consistent. The details of the HCE2 algorithm are as follows.

Definition 1. HCE2 algorithm is comprised of the following algorithms : (HCE2.KeyGen, HCE2.Encrypt, HCE2.TagGen, HCE2.Decrypt).

- *HCE2.KeyGen*$(Para, m_i) \rightarrow k_i$: It is a key generation algorithm that inputs the system parameter $Para$ and data block m_i, and outputs the encryption key k_i.
- *HCE2.Encrypt*$(m_i, k_i) \rightarrow c_i$: It is used to encrypt the original plaintext m_i into ciphertext c_i using the encryption key k_i.
- *HCE2.TagGen*$(c_i) \rightarrow T_i$: It is responsible for calculating the hash value of each ciphertext block c_i to generate the block tag T_i.
- *HCE2.Decrypt*$(k_i, c_i) \rightarrow m_i$: It is used to decrypt the ciphertext c_i into plaintext m_i using the encryption key k_i based on the symmetry of the HCE2 algorithm.

2) Bilinear Maps: A bilinear mapping has an expression of (p, G_1, G_2, e), where p is a large prime number, G_1 and G_2 are two multiplicative cyclic groups of order p, and bilinear map $e : G_1 \times G_1 \rightarrow G_2$ has the following properties:

- *Bilinearity*: $\forall g, h \in G_1, a, b \in Z_p, e(g^a, h^b) = e(g, h)^{ab}$.
- *Non-degeneracy*: $e(g, g) \neq 1$.
- *Computability*: An effective algorithm is available to implement the mapping e.

3) Blockchain and Smart Contract: Blockchain [3] is mainly used to store transaction information and possesses the feature of being tamper-proof. It consists of two parts: blocks and chain structure, where blocks are generated by nodes based on consensus algorithms, and a new block is appended to the previous block to form a chain structure. Each block is marked with the hash value of the previous block, which makes the block information difficult to be tampered with. Each blockchain node holds a copy of the ledger data.

Benefiting from the trusted environment enabled by blockchain technology, smart contracts [24] achieve automatic execution of trusted transactions as part of the Ethereum smart contract system. A smart contract is an agreement that is jointly committed to by the contract participants and can be executed quickly to improve transaction efficiency.

B. System Model

The system model of the scheme includes four entities: *blockchain storage system*, *users*, *smart contracts*, and *system manager*. They have the following definitions:

- *Blockchain Storage System (BSS)*: It is a decentralized storage network based on blockchain consisting of numerous storage nodes (SNs), each of which connects to the blockchain network and provides storage services for users. For uploaded files, BSS arranges several storage nodes for the user to keep two copies of the file, where each storage node stores a partial chunk of the file. Nevertheless, storage nodes are not all honest and may corrupt or delete users' data. Therefore, storage nodes need to periodically complete integrity verification tasks with the assistance of smart contracts. Dishonest storage nodes will be punished, whose bad storage records will be published on the blockchain.
- *User (U)*: Before uploading the data to storage nodes, the user needs to check if the data is already stored, if not, the user is called the initial user (U_1), otherwise, it is called the subsequent user (U_2). U_1 sends the entire encrypted data and authenticators to BSS. U_2 verifies ownership of the data on the ownership contract without uploading the entire data. In addition, data uploaded to the storage nodes are periodically checked for integrity by smart contracts. The audit results are published on the blockchain to avoid repeated audit tasks.
- *Smart Contract (SC)*: A smart contract is a piece of code that executes automatically on the blockchain. The contract execution process is open and transparent, and the results are published in the blockchain ledger. This scheme uses multiple smart contracts to perform a variety of operations, which include user data storage allocation, checking duplicate data blocks, generating challenge values, receiving proof of ownership and proof of integrity, and verifying user ownership and data integrity.
- *System Manager (SM)*: It is used to generate the necessary parameters.

C. Definition

The proposed scheme includes seven algorithms (*Setup*, *U-KeyGen*, *B-KeyGen*, *Encrypt*, *TagGen*, *AuthGen*, *Decrypt*) and two protocols (*Upload*, *Audit*).

1) *Setup*$(\lambda) \rightarrow (Para)$: By inputting a security parameter λ, the algorithm generates the necessary public system parameter $Para$.
2) *UKGen*$(Para) \rightarrow (X_u, PK_u)$: Executed by *SM*, inputting system parameters $Para$, the algorithm outputs the user key pairing (X_u, PK_u).
3) *B-KeyGen*$(m_i) \rightarrow k_i$: Inputting the file block m_i while $F = m_1 \| \cdots \| m_n$, the algorithm returns convergence key k_i.
4) *Encrypt*$(k_i, m_i) \rightarrow c_i$: The algorithm encrypts the plaintext m_i into ciphertext c_i by using the convergence key k_i.
5) *TagGen* contains the following two sub-algorithms:
- *B-TagGen*$(c_i) \rightarrow T_i$: The algorithm inputs ciphertext block c_i to get tag T_i.
- *F-TagGen*$(\{T_i\}) \rightarrow t^*$: The algorithm inputs all block tags $\{T_i\}$ to get the auxiliary file tag t^*.

6) *AuthGen*$(X_u, c_{i,j}) \rightarrow (\sigma_i)$: The algorithm inputs X_u and $c_{i,j}$ to obtain the authenticator σ_i.
7) *Upload*: The user checks if the file F is already stored in BSS before uploading it and executes the following protocol.
- *Initial upload*: If F is not stored, the user executes the above algorithms and following protocol.
- The user first uploads file tag set $T = \{T_i\}$ to check duplicate data blocks.
- SC is used to compare T with the local tags and requires the user to upload the unduplicated block set C_u through file tag set T_u and authenticator set $\{\sigma_i\}$ to SN.
- After receiving C_u and $\{\sigma_i\}$, each SN checks consistency between C_u and T_u, and stores the assigned C_u and $\{\sigma_i\}$.
- *Subsequent upload*: If F has been stored in BSS, U_2 is required to execute PoWs protocol.
- *PoW.Chal*$(Para) \rightarrow chal_x$: The algorithm outputs a challenge value $chal_x$ used for deduplication.
- *PoW.Proof*$(chal_x, \{c_x\}) \rightarrow Proof_x$: Inputting $chal_x$ and the challenged data block set $\{c_x\}$, the algorithm generates a proof of ownership $Proof_x$.
- *PoW.Verify*$(\{\sigma_k\}, Proof_x) \rightarrow 0/1$: Storage nodes generate aggregated authenticators set $\{\sigma_k\}$, based on $Proof_x$ and $\{\sigma_k\}$, the algorithm returns the ownership verification result.

8) *Audit*: The following three algorithms are used for data auditing.
- *Aud.Chal*$(Para) \rightarrow chal_y$: The algorithm inputs $Para$ to get a challenge value $chal_y$ for integrity verification.
- *Aud.Proof*$(chal_y, \{c_y\}, \{\sigma_y\}) \rightarrow Proof_k$: Based on $chal_y$, challenged block set $\{c_y\}$ and authenticator set $\{\sigma_y\}$, the algorithm returns the integrity proof $Proof_k = (\{\mu_k\}, \{\sigma_k\})$, where $\{\mu_k\}$ is generated from storage nodes that own random one copy (copy-1 or copy-2) and $\{\sigma_k\}$ is generated from storage nodes that own the other copy.
- *Aud.Verify*$(Proof_k) \rightarrow (0/1)$: Taking the integrity proof $Proof_k$ as input, the algorithm returns the auditing result.

9) *Decrypt*$(k_i, c_i) \rightarrow m_i$: Owing to the symmetry, the algorithm decrypts the ciphertext c_i to plaintext m_i by using the same convergence key k_i.

D. Security Model

In this study, users and blockchain storage nodes are considered dishonest. Storage nodes may discard or corrupt users' data to save storage resources and provide fake storage proof to

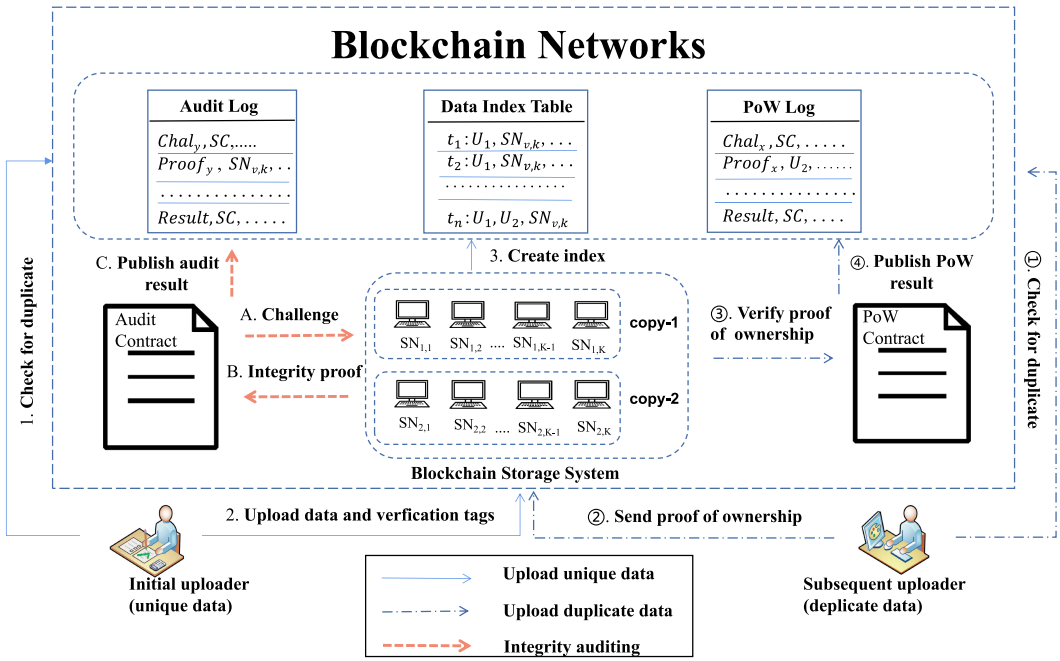


Fig. 1. The overall model of the proposed scheme.

the smart contract. Malicious users may abuse the deduplication mechanism to illegally access the specified data while they do not possess the data.

Definition 2. For the challenge nonce $chal$, no adversary can pass the consistency verification with a non-negligible probability. This study aims to reach the following five security goals.

Data Privacy: After data encryption, our scheme prevents storage nodes from accessing the original plaintext.

Storage Correctness: If the storage nodes pass the integrity verification, it means they must honestly store the user data.

Batch Auditing: We require that smart contracts can perform multiple audit tasks of stored data simultaneously. If a storage node generates a correct response to the integrity challenge, it must honestly store all challenged blocks.

Data Consistency: When the user downloads data that is inconsistent with the original data, it will be unable to get the original plaintext by decrypting the existing ciphertext. In this case, comparing the re-generated tags after decrypting the ciphertext with the original data tags is necessary.

Resistant to Single Point of Failure: The double-copy storage model used in this scheme ensures that the user can still access the specified data in case of single copy failure.

IV. The Proposed Scheme

In this part, we provide a detailed description of the proposed public auditing scheme with secure deduplication for blockchain off-chain storage. Table I presents the necessary notations.

A. Overview

To save storage space and verify the data integrity of storage nodes in the blockchain network, we propose an efficient data auditing scheme with secure deduplication for blockchain storage. For a group of users who want to upload the same data, only the initial user is required to upload the complete data while the other users only execute the PoWs protocol. By using the HCE2 algorithm, users can encrypt the same plaintext into the same ciphertext, thus enabling data privacy protection as well as ciphertext deduplication. Then, to protect the user from losing data under a single point of failure, the ciphertext is stored in a double-copy storage model. If the pre-uploaded data is already stored in storage nodes, the user only needs to provide the SC with the proof of ownership, and if the proof passes, there is no need to upload the entire data. After uploading data, users will regularly verify data integrity through SC used to challenge storage nodes with randomly selected data blocks. Then, storage nodes generate the corresponding proof of integrity and send the proof results to SC. Finally, SC verifies whether the integrity is correct. It should be noted that our scheme can be adapted to multiple copies and the SC can perform data auditing automatically with a public key. The overview of our scheme is shown in Fig. 1.

B. Setup Process

SM executes *Setup*(λ) algorithm to generate system parameters. Then, *SM* chooses two multiplicative cyclic groups G_1 and G_2 with order p to establish a bilinear map $e : G_1 \times G_2 \to G_T$. Moreover, a generator g is selected from G_2 and random elements $u_1, u_2, \ldots, u_s$ are from G_1, two hash functions $H_1 : \{0,1\}^* \to Z_p{}^*, H_2 : \{0,1\}^* \to G_1$, and f is a pseudorandom function $f : \{0,1\}^* \to [1, n]$. Finally, SM releases the system parameters $Para = \{G_1, G_2, e, p, g, u_1, u_2, \ldots, u_s, H_1, H_2, f\}$ on the blockchain.

TABLE I
NOTATIONS

Notation	Description
U_1	Initial user
U_2	Subsequent user
X_u	The user's private key
PK_u	The user's public key
t	The file identifier
λ	Security parameter
G_1, G_2, G_T	Three cyclic multiplicative groups
p	A large prime
g	A generator of G_2
$\{u_j\}$	A random element set of G_1, $j \in [1, s]$
H_1, H_2, f	Three hash functions
$Para$	The public parameter set
n	Number of data blocks
m_i	The plaintext of i-th data block, $i \in [1, n]$
k_i	The convergence key for encrypting m_i to c_i
c_i	The ciphertext of i-th data block
T_i	The tag of i-th ciphertext
σ_i	The authenticator of i-th ciphertext
$SN_{v,k}$	It is the storage node that owns the k-th data chunk of the v-th copy
$2K$	The total number of SN for storing data, where a single copy is stored jointly for every K nodes
c	Number of challenged blocks
τ	Public state information in blockchain
$chal_x$	The ownership challenge message, $chal_x = (c, \tau)$
s_i	The index of challenged blocks
v_i	The coefficient of ownership challenge
$chal_y$	The integrity challenge message, $chal_y = (c, \tau)$
a_i	The integrity challenge index
b_i	The integrity challenge coefficient
$Proof_{v,k}$	Ownership and integrity proof generated by $SN_{v,k}$
$Proof_x$	The proof of ownership by aggregating $proof_k$
$Proof_y$	The proof of integrity by aggregating $proof_k$

C. User Key Generation Process

Based on $Para$, *SM* runs *U-KeyGen* algorithm to generate a private key $X_u \in Z_p$ for a user. Then, *SM* calculates public key $PK_u = g^{X_u}$ and publishes PK_u on the blockchain.

D. Upload Process

When U uploads a file $F = \{m_i\}$ $(i \in [1, n])$, U calculates $t = H_1(F)$ as a file identifier. Next, U checks whether t exists in the data index table on the blockchain. In particular, the data index table is used to store file information including file identifiers, user names and storage node address set. If t does not exist, U will perform the initial upload protocol. Else, U will perform the subsequent upload protocol.

1) Initial Upload: For the file F, U_1 executes the initial upload protocol. The detailed process is as follows.

1) U_1 executes the *B-KeyGen* algorithm to generate block key $k_i = H_1(m_i)$, *Encrypt* algorithm to generate ciphertext $c_i = \mathbf{HCE2.Enc}(k_i, m_i)$, *TagGen* algorithm to generate tag $T_i = H_1(c_i)$. Then, U_1 splits c_i into s segment $c_{i,j} (j \in [1, s])$ and runs *AuthGen* algorithm to generate authenticator $\sigma_i = (M_i \cdot \prod_{j=1}^{s} u_j^{c_{i,j}})^{X_u}$, where $M_i = H_2(t \| i)$.
2) U_1 sends block tag set $T = \{T_i\}$ to BSS for checking duplicate data blocks.
3) SC compares T with the local tag set to return the duplicate tag set T_d and unique tag set T_u. Then, SC requires U to send the unique ciphertext set C_u corresponding to T_u.
4) After receiving the C_u, storage contract checks whether $T_i = H_1(c_i), c_i \in C_u$. If pass, the storage contract generates the second file tag $t^* = H_1(T_1 \| \cdots \| T_n)$ and assigns $\{\sigma = \{\sigma_i\}, C_u\}$ to storage nodes. Here, a file will be stored with two copies. Assume that v-th copy is stored in the storage node list $\{SN_{v,k} | v = 1, 2$ and $k = \{1, \ldots, K\}\}$, where K is practical number of node storing file F and $SN_{v,k}$ is calculated by some distributed storage algorithms, such as DHT, Chord. Table II shows an example of storage relationship. Note that $\{SN_{1,k}\} \cap \{SN_{2,k}\} = \varnothing$, and ρ_k denotes the data block set stored in $SN_{v,k}$.
5) After the storage node stores the corresponding data blocks and authenticators, the data index table will be published on the blockchain for deduplication.
6) After the data index table is created, U_1 removes the file F and authenticator set σ from the local storage.

TABLE II
DATA LOCATION TABLE

Storage node	Data block index
$SN_{1,1} = 3, SN_{2,1} = 7$	$\rho_1 = \{1, 3, 6, 9\}$
$SN_{1,2} = 5, SN_{2,2} = 1$	$\rho_2 = 4, 7, 8\}$
$SN_{1,3} = 4, SN_{2,3} = 9$	$\rho_3 = \{2, 5\}$

2) Subsequent Upload: If file F is duplicated, U_2 will perform a deduplication protocol with SC, detailed deduplication operations are shown as follows.

1) After receiving the upload request from U_2, SC executes *PoW.Chal* algorithm and generates a challenge value $chal_x = (c, \tau)$ for verifying ownership, where $c \in [1, n]$ and τ includes current block hash and timestamp which cannot be controlled by $SN_{v,k}$ and users.
2) After receiving $chal_x$, U_2 executes *PoW.Proof* algorithm to calculate $s_i = f(\tau \| i), v_i = H_1(\tau \| s_i), i \in [1, c]$. Then, U_2 only runs *B-keyGen, HCE2.Enc* to get $k_{s_i} = H_1(m_{s_i})$ and $c_{s_i} = \mathbf{HCE2.Enc}(k_{s_i}, m_{s_i})$. Then, U_2 splits c_{s_i} into $c_{s_{i,j}}$ and calculates $\mu_j = \sum_{i=1}^{c} v_i \cdot c_{s_{i,j}} \bmod p$. Finally, U_2 sends $Proof_x = \{\mu_j\}$ to SC.
3) Besides, SC also calculates the set of the data block indexes $S = \{s_i\}$ and coefficients $V = \{v_i\}$. According to Table II, SC sends $\{S, V\}$ to $SN_{v,k}$. Then, $SN_{v,k}$ calculates $\sigma_k = \prod_{s_i \in \rho_k \cap S} \sigma_{s_i}^{v_i}$ and $Z_k = \prod_{s_i \in \rho_k \cap S} M_{s_i}^{v_i}$. Note that, this part could be executed concurrently in different storage nodes.
4) SC runs the *PoW.Verify* algorithm, if $Proof_x = \{\mu_j\}$ and $PK_u = g^{X_u}$ are correct, we get

$$e(\prod_{k=1}^{K} \sigma_k, g) = e\left(\prod_{k=1}^{K} Z_k \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u\right) \quad (1)$$

If the ownership verification is passed, SC will add U_2 to the data index table in the blockchain. The correctness of the subsequent upload protocol is proved by the following formula.

Algorithm 1. PoW.Verify

Require: Proof of ownership $Proof_x$ and $\{\sigma_k, Z_k\}_{(1\le k\le K)}$
Ensure: Result of ownership proof **re**
Calculate $Left = e(\prod_{k=1}^{K} \sigma_k, g)$
Calculate $Right = e(\prod_{k=1}^{K} Z_k \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u)$
if $Left = Right$ **then**
  Set **re** $= 1$
  U_2 is added to index table on the blockchain
else
  Set **re** $= 0$
end if
return re

Algorithm 2. Aud.Verify

Require: Proof of integrity $\{\mu_{k,j}\}, \{\sigma_k\}$
Ensure: Result of integrity verification **re**
Calculate $\mu_j = \sum_{k=1}^{K} \mu_{k,j}$
Calculate $\sigma = \prod_{k=1}^{K} \sigma_k$
Calculate $Left = e(\sigma, g)$
Calculate $M_{a_i} = H_2(t||a_i)$
Calculate $Right = e(\prod_{i=1}^{c} M_{a_i}^{b_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u)$
if $Left = Right$ **then**
  Set **re** $= 1$
else
  Set **re** $= 0$
end if
return re

Correctness : The correctness analysis of the bilinear pairing in the ownership verification process is as follows.

$$
\begin{aligned}
&e(\prod_{k=1}^{K} \sigma_k, g)\\
&= e(\prod_{k=1}^{K} \prod_{s_i \in \rho_k \cap S} \sigma_{s_i}^{v_i}, g)\\
&= e(\prod_{i=1}^{c} \sigma_{s_i}^{v_i}, g)\\
&= e(\prod_{i=1}^{c} (M_{s_i} \cdot \prod_{j=1}^{s} u_j^{c_{i,j}})^{X_u \cdot v_i}, g)\\
&= e((\prod_{i=1}^{c} M_{s_i}^{v_i}) \cdot (\prod_{i=1}^{c} \prod_{j=1}^{s} u_j^{c_{i,j} \cdot v_i}), g^{X_u})\\
&= e(\prod_{i=1}^{c} M_{s_i}^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u)\\
&= e(\prod_{k=1}^{K} \prod_{s_i \in \rho_k \cap S} M_{s_i}^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u)\\
&= e(\prod_{k=1}^{K} Z_k \cdot \prod_{j=1}^{s} u_j^{\mu_j}, PK_u)
\end{aligned}
$$

E. Integrity Auditing

During the data audit phase, SC executes the following protocols with $SN_{j,k}$ to check data integrity.

1) SC executes the *Aud.Chal* algorithm to generate the challenge value $chal_y = (c, \tau)$, which has same definition as the subsequent upload phase.
2) After obtaining $chal_y$, $SN_{v,k}$ executes the *Aud.Proof* algorithm. Firstly, for $i \in [1, c]$, these nodes calculate $a_i = f(\tau \| i)$, forming $A = \{a_i\}$, where τ cannot be controlled by $SN_{v,k}$, each $SN_{v,k}$ has no way to know this information in advance. For each a_i, $SN_{v,k}$ calculates $b_i = H_1(\tau \| a_i)$. In each audit phase, for two groups of storage nodes $\{SN_{1,k}\}$ and $\{SN_{2,k}\}$, a randomly selected group of nodes will be responsible for generating a linear combination of data blocks $\mu_{k,j} = \sum_{a_i \in \rho_k \cap A} c_{a_i,j} \cdot b_i$, and the other group generates aggregated authenticators $\sigma_k = \prod_{a_i \in \rho_k \cap A} \sigma_{a_i}^{b_i}$.
3) Upon receiving $\{\mu_{k,j}\}$ and $\{\sigma_k\}$, SC executes *Aud.Verify* algorithm to check the correctness of integrity proof.

Correctness: Proof of correctness for integrity verification protocol and ownership verification protocol can be applied mutually. Due to space limitations, it will not be shown here.

F. Decrypt

The user shows $\langle t, t^* \rangle$ before obtaining the original file F from BSS. After downloading the ciphertext $c_1, c_2, \ldots, c_n$, the user performs the following operations.

- Based on ciphertext c_i, the user executes $\mathbf{B-TagGen}(c_i) \rightarrow T_i$ algorithm and generates the block tag T_i.
- Based on the block key k_i and ciphertext c_i, the user runs *Decrypt* algorithm $\mathbf{HCE2.Decrypt}(k_i, c_i)$ to calculate plaintext m_i.
- The user checks whether the equation $t^* = H_1(T_1 \| \cdots \| T_n)$ holds. If pass, the user accepts plaintext m_i, else, rejects it.

G. Batch Auditing

Performing multiple audit tasks simultaneously facilitates audit efficiency. In this subsection, we will describe the batch auditing process in detail. For multiple files uploaded by users, firstly, the contract generates different challenge values based on the total number of blocks of different files. In this case, different files will be one file in the virtual view. Then the contract queries the data distribution table to send the challenge values to the corresponding storage nodes. After receiving the challenge value, the storage node generates the corresponding challenge value index and coefficients according to the challenge value. Then u_k and σ_k are calculated. Finally, send (u_k, σ_k) to the contract. After receiving the parameters, the contract aggregates u_k and σ_k to form U and σ, and calculates M_i. Finally, a bilinear pairing operation is applied to complete the batch audit.

V. Security Analysis

The security analysis of the above security goals is explained in detail below.

Assumption 1. (Divisible Computation Diffie-Hellman (DCDH) problem) Given $g, g^x, g^y \in G_1$, where x and $y \in Z_p$, there is no algorithm that can compute $g^{y/x}$ in polynomial time [43].

Theorem 1. Malicious storage nodes that do not store data honestly are unable to pass the integrity verification during the audit process.

Proof of Theorem 1. After receiving the integrity proof $proof_t = (\sigma_t, \mu_t)$, the contract checks the completeness of the data. We will prove that the adversary cannot pass the integrity

TABLE III
COMPARISONS OF COMPUTATION COSTS

Schemes	Initial upload	Subsequent upload		Data auditing	
		Prove	Verify	Prove	Verify
Scheme [19]	$n_1(1Mul+2Exp)$	$(c_1-1)Mul$	$(2c_1+1)Exp$ $+(2c_1-1)Mul+2Pair$	c_1Exp $+2(c_1-1)Mul$	$(c_1+1)Exp$ $+c_1Mul+2Pair$
Scheme [30]	$n_1(1Mul+2Exp)$ $+n_1E$	\	\	$c_1Exp+1Pair$ $+(2c_1-1)Mul$	$(c_1+3)Exp$ $+(c_1+1)Mul+2Pair$
Our scheme	$n(s+1)(Mul+Exp)$ $+nE$	$s(c-1)Mul$ $+cE$	$(2c+s)Exp$ $+(c+1)Mul+2Pair$	$cExp$ $+c(s+1)Mul$	$(c+s)(Exp+Mul)$ $+2Pair$

$c_1/c=128$.

TABLE IV
COMPARISON OF FUNCTIONS

Operations	Privacy protection	Trusted auditing	Authenticator deduplication	Batch auditing	Data loss prevention	Granularity
Scheme [19]	×	×	✓	✓	×	File-level
Scheme [30]	✓	✓	×	✓	×	Block-level
Our scheme	✓	✓	✓	✓	✓	Block-level

verification through false proof $proof_t{}' = (\sigma_t{}', \mu_t{}')$ without holding the complete data by the following analysis.

$$e(\prod_{i=1}^{c}\sigma_{a_i}^{b_i}, g) = e\left((\prod_{i=1}^{c}M_{a_i}^{b_i})\prod_{j=1}^{s}u_j^{\mu_j}, PK_u\right) \tag{2}$$

As seen from (2), the contract requires to calculate the corresponding $\prod_{i=1}^{c}M_{a_i}^{b_i}$ according to the challenged blocks, which will result in the σ_t' and μ_t' generated by the adversary who does not possess the correct challenged blocks to be detected by the tag binding mechanism.

Moreover, if the adversary wants to achieve $proof_t{}' = proof_t$ with $F' \neq F$, a μ_t' based on $t, c, \tau, Para$ should be solved. Assuming the existence of an algorithm A_1, it is possible to solve for the μ_t' in polynomial time for given $t, c, \tau, Para$. Then, we redesign an algorithm A_2, which can solve the x within given g, g^y, g^{xy} as follows:

1) $\forall \tau \in Z_p$, let $n = c = 1$, $f(\cdot) \equiv 1$, $\sigma_i = g^{xy}$, $g^{X_u} = g^y$, generating the $t, Para$.
2) The μ_t can be solved in the following equation based on algorithm A_1 by using $t, c, \tau, Para$:

$$e(\sigma_1^{b_1}, g) = e\left((M_1^{b_1})\prod_{j=1}^{s}u_j^{\mu_j}, g^{X_u}\right) \tag{3}$$

3) As $f(\cdot) \equiv 1$ and $b_i = f(1, \tau)$, the above equation evolves to

$$e(\sigma_1, g) = e\left(M_1\prod_{j=1}^{s}u_j^{\mu_j}, g^{X_u}\right) \tag{4}$$

4) Compute $\alpha = M_1\prod_{j=1}^{s}u_j^{\mu_j}$.
5) Note that $\sigma_i = g^{xy}, g^{X_u} = g^y$, we thus get

$$e(g^{xy}, g) = e(\alpha, g^y) \tag{5}$$

6) By the properties of bilinear mapping, we get

$$\alpha = g^x \tag{6}$$

The g^x can be solved by the algorithm A_2 in a polynomial time by using g, g^y, g^{xy}, which is in contradiction to Assumption 1. Thus, the proof is finished.

Theorem 2. This scheme ensures that the storage nodes store data honestly and that the results of the verification are true.

Proof of Theorem 2. The proposed scheme achieves trusted auditing based on smart contract. This means that the proof and verification results generated during the auditing process can be shared among users for rechecking.

Theorem 3. Data privacy can be protected in the proposed scheme.

Proof of Theorem 3. Based on the MLE (HCE2) [12], the proposed scheme enables the user data with PRV$-CDA level security. Only the user who has the original plaintext can decrypt the ciphertext using the key associated with the plaintext. Thus, it is impossible for storage nodes and malicious adversaries to retrieve plaintext even if they have gained the encrypted data illegally.

Theorem 4. The proposed scheme ensures that users can still access the data in case of a single point of failure.

Proof of Theorem 4. Benefit from blockchain, the storage node can offer more storage space to the users. Based on this, the proposed scheme distributes the double-copy data in several storage nodes which is more effective to increase the fault tolerance of data in case of a single point of failure.

VI. Performance Evaluation

In this section, we analyze the performance of our scheme and compare it with two related schemes [19], [30] in the following aspects: functionality, computational overhead and storage overhead.

A. Experiment Environment

In this part, we implement the proposed scheme based on BN256 curve with 128-bit security in the Ethereum platform and Golang language. The test environment is Intel(R) Core(TM) i5-7400 CPU 3.00 GHz 16 GB RAM, Windows 10. In particular, we use SHA-256 to generate the convergent key and AES-128 to encrypt users' data in the HCE2 algorithm. We set that the smart contract will verify each user's uploaded data once in an audit period.

We first compare our scheme with compared schemes theoretically. According to Ng et al.'s work [44], 4 KB is the most suitable block size for gaining maximum space-saving efficiency of the deduplication, we set the data block size to 4 KB, each block is divided into 128 sectors, each sector is exactly 256 bits in size, which is same as the block size in [19], [30]. Assume that the number of blocks in the compared scheme is n_1, thus, $n_1/n = 128$. c denotes the number of challenged blocks. Due to the negligible overhead of the hash operation, we will not discuss it. We use E to show an encryption operation in HCE2, Mul to show a multiplication operation, Exp to show an exponent operation, and $Pair$ to show a bilinear map. Table III shows the computational cost of [19], [30] and our scheme.

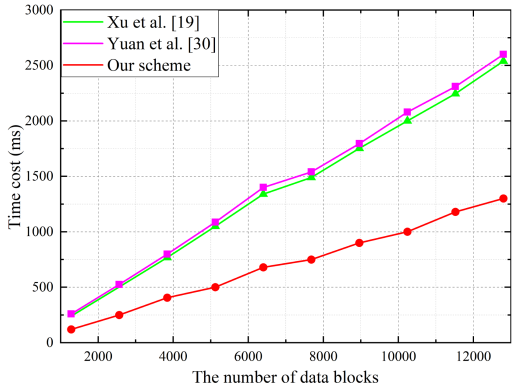
B. Comparison of Functions

As shown in Table IV, we show the comparison results of functions. The notion of "Data loss prevention" demonstrates whether compared schemes can protect users from data loss under a single point of failure, "Authenticator deduplication" indicates whether these schemes can achieve both data deduplication and authenticator deduplication. In addition, the notion of "Trusted auditing" represents whether those schemes can eliminate untrusted third parties and achieve trust audit. Additionally, Yuan et al.' scheme [30] and our scheme achieve encrypted data upload, which ensures data privacy. The above schemes all achieve batch auditing. Meanwhile, compared to Xu et al.' scheme [19], "block-level" properties save space and bandwidth effectively. In summary, our scheme is more efficient than compared schemes.

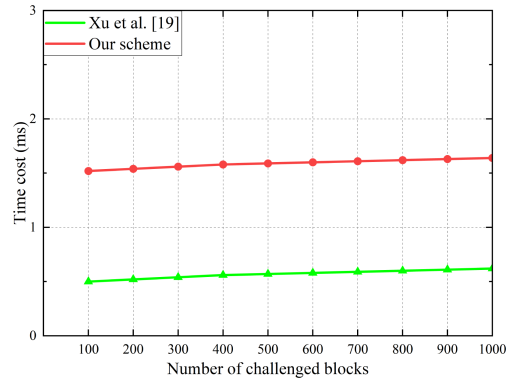
C. Computation Overhead

In this subsection, we conduct a series of operations to test the computation costs that are performed off-chain and on-chain of all compared schemes, respectively.

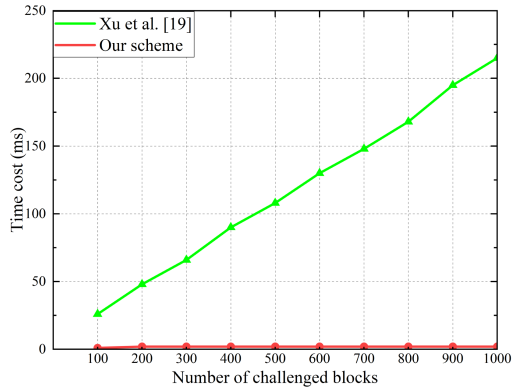
1) Off-Chain Part Costs: In this part, we evaluate the off-chain computation overhead with compared schemes. Fig. 2 contains the four off-chain operations: initial upload, ownership proof generation, integrity proof generation and ownership proof verification. For a clearer description, we take the number of



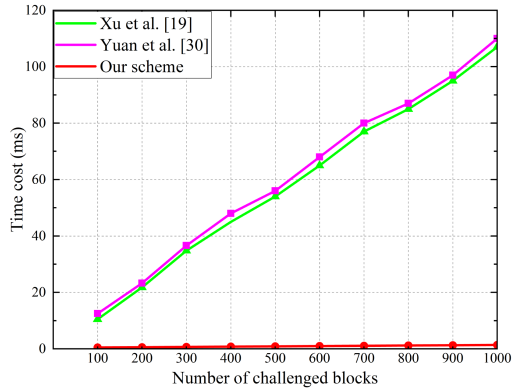
(a) Initial upload



(b) Ownership proof

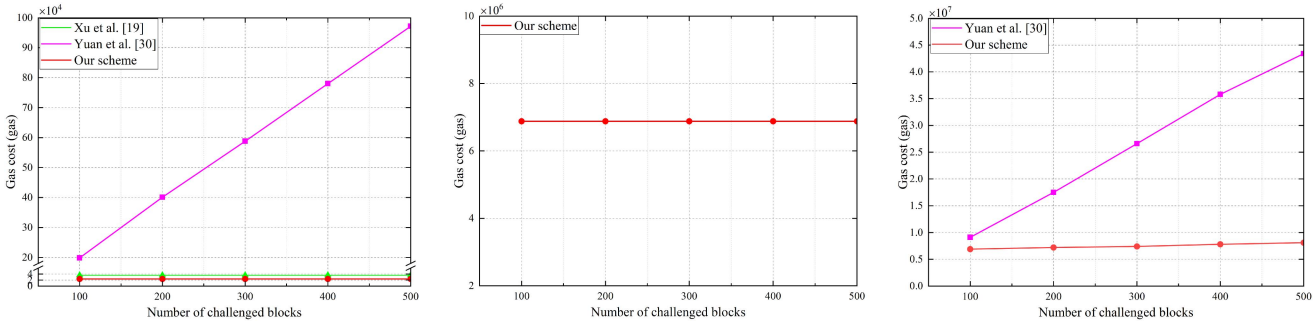


(c) Ownership verification



(d) Integrity proof

Fig. 2. Comparisons of off-chain costs.



(a) Gas cost of challenge (b) Gas cost of ownership verification (c) Gas cost of integrity verification

Fig. 3. Comparisons of on-chain costs.

blocks of [19] as the benchmark. Fig. 2(a) shows the computational overhead in the initial upload phase. We can see that the time cost increases roughly linearly for all schemes, where the time cost of [19] is 251 ms for 1280 blocks, 1319 ms for 6400 blocks, and 2531 ms for 12800 blocks. Scheme [30] uses encrypted data to generate the authenticator and the total overhead is larger than scheme [19], 264 ms for 1280 blocks, 1346 ms for 6400 blocks and 2598 ms for 12800 blocks, respectively. From Table IV, our scheme needs fewer exponential operations to generate authenticators, The final overhead is 131 ms for 1280 blocks, 659 ms for 6400 blocks and 1320 ms for 12800 blocks.

Scheme [30] essentially does not distinguish between initial and subsequent uploads, and performs the same operation for each data upload. Therefore, only the computational overhead of scheme [19] and our scheme will be shown in Fig. 2. As can be seen in Fig. 2(b), our scheme takes more time to generate the ownership proof than [19], which is because our scheme implements encrypted data storage to protect the privacy and needs to encrypt the data first in each generation of proof of ownership.

From Fig. 2(c), due to fewer exponential and multiplicative operations, the overhead of our scheme is much lower than that of [19]. Specifically, the cost of [19] is 65.46 ms for 300 challenged blocks and 98.48 ms for 460 challenged blocks, while our scheme consistently consumes 1 ms.

Furthermore, Fig. 2(d) shows the total time cost comparison of integrity-proof generation. For 300 challenged blocks, the computation costs of two schemes [19], [30] and our scheme are 32.9 ms, 33.9 ms, and 0.7 ms. For 460 challenged blocks, the computation costs increase to 47.9 ms, 53.3 ms, and 0.8 ms, respectively.

2) On-Chain Part Costs: We evaluate the on-chain overhead by testing the gas consumption. Gas is the unit used to measure workload on Ethereum. All kinds of transactions, storage, and other operations generated on Ethereum require gas to drive the Ethereum Virtual Machine (EVM) to work. That is to say, the simpler an operation is, the less gas consumes. We measure each on-chain operation of compared schemes 10 times to get average gas consumption.

The comparison of the on-chain gas overhead is shown in Fig. 3, containing three operations: challenge value generation, ownership verification and integrity verification. From Fig. 3, it is clear that for the same number of challenge blocks, our scheme generates much less gas than [30]. Scheme [19] sends a constant challenge value during each challenge and the overhead remains the same. When challenging 300 blocks, the gas cost are 2.91×10^4 gas, 5.9×10^5 gas and 2.61×10^4 gas, respectively. In our scheme, the ownership verification final operation is performed by the smart contract, and the generated overhead is independent of the number of challenge blocks, which is always maintained at 6.88×10^6 gas.

According to Ateniese et al. [5], if the total number of data blocks is 10000 and the damaged rate of outsourced data block is 1%, the precision rates of discovering the malicious behavior of $SN_{v,k}$ are 95% for 300 challenged blocks, and 99% for 460 challenged blocks. From Fig. 3(c), the gas cost of our scheme does not increase significantly with the increase of challenged blocks, The major cause is that our scheme does not require signature verification and excessive exponentiation operations. The integrity verification costs of our scheme are 0.71×10^7 gas for 300 challenged blocks and 0.73×10^7 gas for 460 challenged blocks, while Yuan et al.' scheme [30] consumes 2.63×10^7 gas and 4.04×10^7 gas, respectively.

TABLE V
NOTATIONS IN EVALUATION

Notations	Description
n_1	Total number of file blocks in [19], [30]
n	Total number of file blocks in our scheme
n_u	Number of user who own file F
d	Number of duplicate blocks of F in BSS
$\|F\|$	Length of the outsourced file F
$\|G\|$	Length of the element in G

D. Storage Overhead

In this part, we evaluate the storage overhead of SN. For a clearer expression, some necessary notations are shown in Table V. Concerning storage cost, [30] adopts server-side deduplication, for users with the same file F, storage nodes are required to store $n_u(n_1|G| + 1|F|)$. Since [19] and our scheme implement data and authenticator deduplication, only the initial user needs to upload the complete file and authenticators. In addition, we also implement block-level data elimination, so the final storage overheads for [19] and our scheme are

$n_1|G| + 1|F|$ and $n|G| + \frac{n-d}{n}|F|$, respectively. Moreover, as the data duplication rate $\frac{d}{n}$ increases, the storage overhead in the scheme will further decrease.

VII. Conclusion and Future Work

Focusing on data redundancy and integrity issues in blockchain off-chain storage, we propose a public auditing scheme with secure deduplication. Based on the message-locked encryption and improved authenticator generation algorithms, our scheme achieves both encrypted data and authenticator deduplication to save storage resources. Using smart contracts, bilinear pairings, and a double-copy storage model, we effectively implemented public data auditing while protecting user data from a single point of failure in distributed storage. We demonstrated that our scheme can achieve the desired security goals and give detailed experimental results. The performance evaluation suggests that the proposed scheme is efficient. In the future, we will study the issues of efficient user revocation and fair arbitration under this model.

Acknowledgments

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

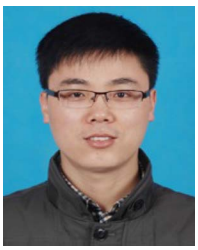
References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, vol. 4, 2008, Art. no. 21260.
[2] M. Swan, *Blockchain: Blueprint for a New Economy*. California, USA: O'Reilly Media, Inc., 2015.
[3] Z. Zheng, S. Xie, Hong-Ning Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, 2018.
[4] N.Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *J. Netw. Comput. Appl.*, vol. 162, 2020, Art. no. 102656.
[5] G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2017, pp. 598–609.
[6] A. Juels and B. S. Kaliski Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.
[7] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. cryptol. Inf. Secur.*, Springer, 2008, pp. 90–107.
[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2010.
[9] B. K. Pavan Mahesh, S. K. Ramasubbareddy, and E. Swetha, "A review on data deduplication techniques in cloud," *Embedded Syst. Artif. Intell.*, vol. 1076, pp. 825–833, 2020.
[10] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. IEEE 22nd Int. Conf. Distrib. Comput. Syst.*, 2002, pp. 617–624.
[11] Y. He, H. Xian, L. Wang, and S. Zhang, "Secure encrypted data deduplication based on data popularity," *Mobile Netw. Appl.*, vol. 26, no. 4, pp. 1686–1695, 2021.
[12] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2013, pp. 296–312.
[13] Y. Wang, M. Miao, J. Wang, and X. Zhang, "Secure deduplication with efficient user revocation in cloud storage," *Comput. Standards Interfaces*, vol. 78, 2021, Art. no. 103523.
[14] S. Li, C. Xu, Y. Zhang, Y. Du, and K. Chen, "Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 16, no. 1, pp. 134–146, Jan./Feb. 2022.
[15] G. Zhang, H. Xie, Z. Yang, X. Tao, and W. Liu, "BDKM: A blockchain-based secure deduplication scheme with reliable key management," *Neural Process. Lett.*, vol. 54, no. 4, pp. 2657–2674, 2022.
[16] Q. Hu, "A data integrity verification scheme of deduplication for cloud ciphertexts," in *Proc. IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf.*, 2020, pp. 865–869.
[17] W. Shen, Y. Su, and R. Hao, "Lightweight cloud storage auditing with deduplication supporting strong privacy protection," *IEEE Access*, vol. 8, pp. 44359–44372, 2020.
[18] S. Li, C. Xu, Y. Zhang, A. Yang, X. Wen, and K. Chen, "Blockchain-based efficient public integrity auditing for cloud storage against malicious auditors," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Springer, 2021, pp. 202–220.
[19] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1421–1432, Jul./Sep. 2020.
[20] X. Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, "One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[21] H. Zhao, X. Yao, X. Zheng, T. Qiu, and H. Ning, "User stateless privacy-preserving TPA auditing scheme for cloud storage," *J. Netw. Comput. Appl.*, vol. 129, pp. 62–70, 2019.
[22] Y. Yang, Y. Chen, F. Chen, and J. Chen, "An efficient identity-based provable data possession protocol with compressed cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1359–1371, 2022.
[23] P. Wang, X. Su, M. Jourenko, Z. Jiang, M. Larangeira, and K. Tanaka, "Environmental adaptive privacy preserving contact tracing system: A construction from public key rerandomizable bls signatures," *IEEE Access*, vol. 10, pp. 37181–37199, 2022.
[24] G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.
[25] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 923–937, 2019.
[26] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Inf. Sci.*, vol. 519, pp. 348–362, 2020.
[27] J. Li, J. Wu, G. Jiang, and T. Srikanthan, "Blockchain-based public auditing for Big Data in cloud storage," *Inf. Process. Manage.*, vol. 57, no. 6, 2020, Art. no. 102382.
[28] F. Kefeng, L. Fei, Y. Haiyang, and Y. Zhen, "A blockchain-based flexible data auditing scheme for the cloud service," *Chin. J. Electron.*, vol. 30, no. 6, pp. 1159–1166, 2021.
[29] 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 Trans. Serv. Comput.*, vol. 13, no. 2, pp. 289–300, Mar./Apr. 2019.
[30] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, 2020.
[31] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1210–1223, May 2021.
[32] B. Li et al., "Cooperative assurance of cache data integrity for mobile edge computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4648–4662, 2021.
[33] B. Li, Q. He, L. Yuan, F. Chen, L. Lyu, and Y. Yang, "EdgeWatch: Collaborative investigation of data integrity at the edge based on blockchain," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 3208–3218.
[34] K. Vijayalakshmi and V. Jayalakshmi, "Analysis on data deduplication techniques of storage of Big Data in cloud," in *Proc. IEEE 5th Int. Conf. Comput. Methodol. Commun.*, 2021, pp. 976–983.
[35] S.S. Tyagi et al., "An analysis and comparative study of data deduplication scheme in cloud storage," in *Proc. Mach. Learn. Predictive Anal.*, Springer, 2021, pp. 423–431.
[36] H. Gajera and M. L. Das, "Fine-grained data deduplication and proof of storage scheme in public cloud storage," in *Proc. IEEE Int. Conf. Commun. Syst. Netw.*, 2021, pp. 237–241.
[37] M. Miao, G. Tian, and W. Susilo, "New proofs of ownership for efficient data deduplication in the adversarial conspiracy model," *Int. J. Intell. Syst.*, vol. 36, no. 6, pp. 2753–2766, 2021.

[38] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proc. 2nd ACM Conf. Data Appl. Secur. Privacy*, 2012, pp. 1–12.
[39] G. Tian et al., "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3941–3954, Nov./Dec. 2022.
[40] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of IoT data," in *Proc. Cloud Comput. Secur. Workshop*, 2017, pp. 45–50.
[41] S. Ruj, M. S. Rahman, A. Basu, and S. Kiyomoto, "BlockStore: A secure decentralized storage framework on blockchain," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl.*, 2018, pp. 1096–1103.
[42] S. Vimal and S. K. Srivatsa, "A new cluster P2P file sharing system based on IPFS and blockchain technology," *J. Ambient Intell. Humanized Comput.*, vol. 23, pp. 1–7, 2019.
[43] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.*, Springer, 2003, pp. 301–312.
[44] C.-H. Ng and P. P. C. Lee, "RevDedup: A reverse deduplication storage system optimized for reads to latest backups," in *Proc. 4th Asia-Pacific Workshop Syst.*, 2013, pp. 1–7.

Qingyang Zhang received the BEng degree and PhD degree in computer science from Anhui University, in 2021. He is currently a Lecturer of School of Computer Science and Technology with Anhui University. His research interest includes edge computing, computer systems, and security.

Dongfang Sui is now a research student in the School of Computer Science and Technology, Anhui University. His research focuses on the security of Blockchain.



Jie Cui (Senior Member, IEEE) received the PhD degree in University of Science and Technology of China, in 2012. He is currently a professor and PhD supervisor of the School of Computer Science and Technology with Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has more than 150 scientific publications in reputable journals (e.g. *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, *IEEE Transactions on Cloud Computing*, and *IEEE Transactions on Multimedia)*, academic books and international conferences.

Chengjie Gu received the PhD degree in the Nanjing University of Posts and Telecommunications, in 2012. From 2012 to 2017, he was an innovation team leader in the 38th Research Institute of CETC and conducted research and development in the communication and networking sector. Currently he is the president of security research institute in new H3C group. He is also studying for postdoctoral fellowship with the USTC. He is a high-level innovation leader of Anhui province and a cybersecurity expert of Zhejiang province in China. His research interest includes network security and trusted network architecture, etc.

Hong Zhong (Member, IEEE) received the PhD degree in computer science from University of Science and Technology of China, in 2005. She is currently a professor and PhD supervisor of the School of Computer Science and Technology with Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). She has over 200 scientific publications in reputable journals (e.g. *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, and *IEEE Transactions on Big Data*), academic books and international conferences.