Blockchain-Based Shared Data Integrity Auditing and Deduplication

Ying Miao, Keke Gai, *Senior Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Jaideep Vaidya, *Fellow, IEEE*

***Abstract*—Data deduplication and integrity auditing based blockchain plays an important role in guaranteeing secure and efficient cloud storage services. However, existing data deduplication schemes support auditing either with the assistance of a trust center (key server or third-party auditor) or bear the waste of computation and storage resources caused by repetitive authenticators storage and key storage. In this paper, we propose a blockchain-based shared data integrity auditing and deduplication scheme. Specifically, we propose a deduplication protocol based on ID-based broadcast encryption without key servers and achieve key deduplication on the user side. Next, we propose a data integrity auditing protocol by using the characteristic of convergent encryption to achieve authenticator deduplication on the cloud service provider side. Besides, we achieve decentralized data integrity auditing based blockchain without relying on a single trusted third-party auditor and improve the credibility of the auditing result. On this basis, we propose two bath auditing protocols for different scenarios to improve efficiency. Security and performance analysis demonstrates that the authenticators' storage cost on the cloud storage provider side can be reduced from $\mathcal{O}(\mathcal{F})$ to $\mathcal{O}(1)$ and the key storage cost on the user side can be reduced from $\mathcal{O}(\mathcal{F})$ to $\mathcal{O}(1)$ as well.**

***Index Terms*—Data deduplication, key deduplication, tag deduplication, batch auditing, blockchain.**

I. Introduction

DATA deduplication with integrity auditing ensures the reliability of cloud storage services, especially as the demand for cloud storage surges today. According to recent studies estimated by *Internet Data Center* (IDC),[1] the data held by a single person reached 5200 GB. Storing these huge data locally is difficult. Hence, cloud storage services attract users from individuals and corporations, due to their merits of cost savings, easy sharing and access. A number of *Cloud Service Providers* (CSPs) have emerged in recent years, like Google Inc., Oracle, Tencent cloud, etc. According to IDC's report in 2021,[2] the global IaaS market reached 64.3 billion in 2020. However, data loss incidents occurred from time to time. For example, accenture fell prey to a LockBit ransomware attack in 2021. The culprits claimed to have stolen 6 TB worth of data, for which they requested a ransom of 50 million.[3] Although many data deduplication and integrity auditing schemes have been proposed in recent years to solve this problem, these schemes still suffer from excessive storage overhead problems, including both on the user side and the server side.

To protect data integrity, data integrity auditing techniques were born. The technology allows for checking integrity without downloading the whole data. Up to now, many data auditing schemes [1], [2], [3] have been proposed. In data integrity auditing schemes, for the convenience of users, a trusted *Third-Party Auditor* (TPA) is usually introduced to assist the auditing process. Thus, the TPA plays a key role in the auditing model. However, when the TPA is compromised, the auditing results are not reliable. Blockchain technology dates back to Satoshi Nakamoto's bitcoin [4], which is essentially a decentralized database that is collectively maintained through all nodes. Blockchain has outstanding characteristics in decentralization, immutability, and traceability [5], [6], [7]. Blockchain technology provides a good solution to improve the credibility of TPA. Many researchers have focused on blockchain-based data integrity auditing in recent years. Xu et al. [8] proposed a blockchain-enabled deduplicate data auditing mechanism. Their scheme utilized the blockchain to supervise the TPA. Yuan et al. [9] proposed a blockchain-based public auditing and secure deduplication with fair arbitration. Their scheme utilized smart contracts to perform data auditing and achieve fair arbitration. Similarly, Tian et al. [10] proposed a blockchain-based secure deduplication and shared auditing scheme in decentralized storage environment. Their scheme unified the authenticator and made it the same for the same file to save storage costs on the CSP side. The main challenge in these schemes is to improve

Manuscript received 14 March 2023; revised 1 November 2023; accepted 12 November 2023. Date of publication 28 November 2023; date of current version 11 July 2024. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. This work was supported in part by the National Key Research and Development Program of China udner Grant 2021YFB2701300 and in part by the National Natural Science Foundation of China under Grants 62372044 and 62232002. *(Corresponding author: Keke Gai.)*

Ying Miao, Keke Gai, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3120225470@bit.edu.cn; gaikeke@bit.edu.cn; liehuangz@bit.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Jaideep Vaidya is with Management Science and Information Systems Department, Rutgers University, Newark, NJ 07102-1803 USA (e-mail: jsvaidya@business.rutgers.edu).

Digital Object Identifier 10.1109/TDSC.2023.3335413

[1]https://www.computerworld.com/article/2493701/by-2020--there-will-be-5-200-gb-of-data-for-every-person-on-earth.html

[2]https://www.gartner.com/en/newsroom/press-releases/2021-06-28-gartner-says-worldwide-iaas-public-cloud-services-market-grew-40-7-percent-in-2020

[3]https://www.cybertalk.org/2022/04/26/top-5-cloud-security-breaches-and-lessons/

the auditing efficiency. In general, the auditor should deal with the transactions from the same DO and different DOs. Dealing with the auditing transactions one by one is inefficient. How to improve the auditing efficiency in blockchan system is a problem.

However, the existing schemes still have disadvantages in heavy storage costs on both the CSP side and the user side. In general, the data owner should split the file into a series of blocks and then generate the authenticator tags for data integrity checking. This method is not suitable for the scenario where many users have the same file, because each user has its own tag, and users' tags are different. To cooperate with the user to complete the integrity check, the CSP must store all of these tags for all users. This method causes a waste of storage resources. Hence, it is of great significance to find a method to achieve tag deduplication and satisfy the data integrity auditing requirements of users at the same time.

Besides, as the number of cloud storage users increases, the amount of data grows in an explosive way. To achieve data deduplication, *Proof of Ownership* (PoW) method was introduced [11]. PoW adopts the challenge-response mode to prove the data possession. Another important technology is *Convergent Encryption* (CE) [12]. In CE, the convergent key is usually derived from the hash value of the data. The data is encrypted by the convergent key. Thus, the same data corresponds to the same ciphertext. According to this property, the CSP can execute data deduplication on ciphertexts. To resist key guessing attack for low entropy message, *Randomized Convergent Encryption* (RCE) was introduced [11], which utilized a random key k_l to encrypt the message (e.g. $e_1 = E_{k_l}(M)$), and then the random key was encrypted by the convergent key (e.g. $e_2 = k_l \oplus K_c$). Thus, the data can be recovered by only holding the convergent key. However, in order to obtain the plaintext, the data owner should store the convergent keys for each file. If the data owner has massive data, the data owner bears the burden of storing many convergent keys. To overcome the burden of key storage, some data deduplication schemes [13], [14], [15] introduce a key server to assist in recovering the convergent key. However, this approach requires frequent interaction with the key server, especially for frequently used data. Besides, a single key server easily suffers from a single point of failure when the key server is compromised. Some schemes [16] utilize multiple key servers to replace a single server, but it still requires frequent interaction with the key servers. When the data are used more frequently, it results in higher computation costs associated with interacting with the key servers.

Nowadays, some studies have focused on data deduplication and integrity auditing at the same time [8], [9], [17]. However, most of these schemes mainly focus on data deduplication and data integrity auditing. Few schemes consider tag deduplication and key deduplication on the user side when a data owner has massive data. Key deduplication can be accomplished through a key negotiation protocol between the DO and the CSP. It's important to note that there is no direct interaction required between different users in this process. This assumption is rational because subsequent uploaders don't need to re-upload the same data, especially when the amount of data is large, which not only saves a significant amount of upload time but also contributes to the overall efficiency of the system.

To reduce the storage overhead both on the user side and the CSP side at the same time, it is important to design a data integrity auditing and deduplication scheme supporting tag deduplication and key deduplication. The contributions of this work can be summarized as follows:

1) Focus on key storage deduplication, we utilize *ID-based broadcast encryption* (IBBE) to achieve convergent key management. The proposed scheme does not rely on key servers or any trusted third parties. Instead of storing many convergent keys for each file, the DO who only stores the secret key and the IBBE key can recover its data in our scheme. The proposed protocol can achieve efficient space-saving on the user side.
2) Focus on authenticator storage deduplication, we utilize *Boneh-Lynn-Shacham* (BLS) signature and the characteristic of convergent encryption to achieve lightweight authenticator generation. Thus, the users own the same file and the authenticator at the same time. The CSP only needs to store the data once.
3) Focus on improving the credibility of the auditing results, the proposed scheme does not rely on a centralized TPA instead of a consensus mechanism to achieve data integrity auditing, which can avoid a single point of failure and improve the credibility of the auditing results. Different from previous schemes, to improve efficiency, we propose two efficient batch auditing protocols for different transactions on the blockchain. One is one user with multiple files auditing, and the other is many users with the same file auditing.
4) Security analysis shows that the proposed scheme can achieve security goals. Experiments demonstrate that the proposed scheme is efficient in many ways, including the computation costs of authenticators, the storage overhead on the user side and CSP side.

The remainder of this paper is organized as follows: Section II introduces the related work. Section III introduces the basic knowledges. Section IV presents the problem formulation. Section V presents the construction of the proposed scheme. Section VI shows the correctness and security analysis. Section VII shows theoretical analysis and performance analysis. Finally, Section VIII draws conslusions of the paper.

II. Related Work

A. Data Integrity Auditing

Ateniese et al. [19] proposed the concept of *Provable Data Possession* (PDP), which utilized a random sampling technique to check the outsourced data. By using this technique, the status of the whole data can be known by the user in time. Afterward, many public auditing schemes supporting different functions have been proposed, such as certificate-based auditing [20], identity-based auditing [1], certificateless auditing [2], multi-cloud auditing [21], privacy-preserving auditing [22], group data auditing [23], etc. However, most of these TPA-assisted public auditing schemes exist a problem: overreliance on TPA.

TABLE I
COMPARISON OF DATA INTEGRITY AUDITING AND DATA DEDUPLICATION SCHEMES (PARTIAL)

Scheme	Data auditing			Data deduplication			
	Batch Auditing	Decentra-lization	Authenticator deduplication	No key server	No KMP	Model	Granularity
[8]	√	×	×	√	×	Client-side	File-level
[9]	√	√	×	√	×	Server-side	File-level
[10]	√	√	√	√	×	Client-side	File-level
[18]	√	×	×	√	×	Client-side	File-level

NO KMP: no MLE key management problem.

Decentralization technology brings many benefits, some decentralized data integrity auditing schemes have been proposed. Miao et al. [24] proposed a decentralized and privacy-preserving data integrity auditing scheme. Yang et al. [25] proposed a data integrity auditing scheme in multi-replica and multi-cloud environment. Jiang et al. [26] proposed a data integrity auditing scheme by utilizing *Identity-Based Cryptography*. These schemes [24], [25], [26] utilized public information on the blockchain to generate challenge information, which reduced the power of TPA. Shu et al. [27] utilized blockchain network to replace the centralized TPA, which mitigated the influence of untrust auditors. Li et al. [28] randomly chose DOs from the blockchain network as a group to act the role of TPA, which reduced the probability of the compromised auditor. However, most of these schemes [27], [28] can not support authenticators' deduplication and batch auditing.

B. Data Deduplication

The concept of data deduplication was proposed by Douceur et al. [12], in which *Convergent Encryption* (CE) was proposed. Afterward, many CE variants have been proposed. Bellare et al. [11] formalized CE and its variants as a cryptographic primitive of "Message-Locked Encryption"(MLE). However, it does not suit for the low-entropy data. Because the MLE algorithm easily suffers from brute-force attack. To resist this attack, some schemes introduce a key server to help to generate MLE keys. Zhang et al. [29] utilized ring key from JointCloud system to implement data deduplication. Zheng et al. [14] utilized server-aided MLE to achieve media data deduplication. However, this method is faced with the single point of failure. Yan et al. [30] achieved cloud data deduplication with access control and improved the flexibility. Yu et al. [31] achieved duplication check and flexible tag generation for integrity check and they utilized proxy re-encryption to unify the data format.

Subsequent schemes [16] utilized a group of key servers. However, the key problem of trusting the whole key servers still exists. To reduce the excessive dependence on the server, Zhang et al. [29] proposed a proactivization mechanism to replace the group members periodically. To remove the key servers and reduce the key management cost, Liu et al. [32] utilized IBBE to negotiate a key among users who own the same file. Yang et al. [33] introduced a tunable encrypted deduplication primitive, which offered a flexible mechanism for adjusting the tradeoff between storage efficiency and data confidentiality.

C. Data Auditing With Deduplication

Some related work in terms of data integrity auditing with deduplication was compared in Table I. Yuan et al. [34] first proposed data integrity auditing scheme with deduplication. Their scheme achieved data deduplication and authentication tags deduplication. However, the authentication tags should be generated and uploaded by all users for an identical file, which leads to high computational costs. Xu et al. [8] utilized blockchain to supervise a not fully trusted TPA. However, their scheme performs deduplication and auditing over plaintext. Bai et al. [18] utilized IBBE to reduce the storage overhead of the user side. Yuan et al. [9] utilized the smart contracts to assist the auditing and achieve fair arbitration. However, the number of authenticators in [9], [18] is linear to the number of users and files, which leads to a huge storage burden for the CSP side. To overcome this issue, Tian et al. [10] unified the authenticator and made it the same for the same file. But the user still needs to store key for each file. Thus, the storage overhead still relates to the number of files.

To reduce the computation overhead on the user side, Li et al. [35] utilized a trusted proxy server to help generate the authentication tags. However, it is a strong assumption that the proxy is fully trustworthy. Afterwards, Liu et al. [36] proposed a scheme without the trusted proxy server. Their scheme achieved tags deduplication with the help of MLE key. However, their scheme cannot guarantee the auditing result when data are low-entropy. To solve this problem, Gao et al. [17] utilized the random key chosen by the initial uploader to replace the MLE key. However, how to guarantee the reliability of the subsequent uploader's auditing results is a problem. Recently, Zhang et al. [37] introduced an integrity auditing scheme that incorporates client-side deduplication. This feature enables the initial uploader to upload both the data and authenticators. However, it's important to note that the Proof of Work (PoW) mechanism in this scheme only verifies the data integrity of the initial uploader and doesn't extend to other users.

III. PRELIMINARIES

A. Bilinear Pairing

Suppose $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic multiplicaive groups with the same order q. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfies the following three properties:

1) *Bilinearity:* for any $u, v \in \mathbb{G}$ and any $a, b \in \mathbb{Z}_q^*$, there is $e(u^a, v^b) = e(u, v)^{ab}$.
2) *Computability:* $\forall s, t \in \mathbb{G}$, there exists an efficient algorithm to retrieve $e(s, t) \in \mathbb{G}_T$ in polynomial time.
3) *Non-degeneracy:* for some $g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2) \neq 1$.

B. Hard Problems

Let $\mathbb{G} = \langle g \rangle$ be a multiplicative cyclic group with prime order q. Let two random elements be $a, b \in \mathbb{Z}_q^*$.

Definition 1: (*Computational Diffe-Hellman* (CDH) Assumption) Given $(g, g^a, g^b \in \mathbb{G}_1)$, the CDH assumption holds if there is no *probabilistic polynomial-time* (PPT) adversary $\mathcal{A}$ which can retrieve value g^{ab} with non-negligible probability. That is, the probability

$$\Pr[g^{ab} \leftarrow \mathcal{A}_{CDH}\left(g, g^a, g^b\right) : a, b \leftarrow \mathbb{Z}_q^*]$$

is negligible.

Definition 2: (*Discrete Logarithm* (DL) Assumption) Given $(g, g^a \in \mathbb{G}_1)$, the DL assumption holds if there is no *probabilistic polynomial-time* (PPT) adversary $\mathcal{A}$ which can retrieve the value a with non-negligible probability. That is, the probability

$$\Pr[a \leftarrow \mathcal{A}_{DL}\left(g, g^a\right) : a \leftarrow \mathbb{Z}_q^*]$$

is negligible.

C. Identity-Based Broadcast Encryption (IBBE)

In *Identity-Based Broadcast Encryption* (IBBE), a set of identities participate in the key negotiation. When a message is encrypted by the nogotiated key, all identities in the set can decrypt the message. In this paper, we adopt an IBBE scheme [38] to encrypt convergent keys. The IBBE scheme includes four schemes. For the convenience, the notion $\mathcal{IB}$ represents IBBE.

- $\mathcal{IB}.Setup(\lambda, u)$: by taking a security parameter λ and a number of users u as input, the algorithm outputs the master key msk and public parameters pp.
- $\mathcal{IB}.Extract(ID_i, msk)$: by taking a user's identity ID_i and the master secret key msk as input, the algorithm outputs the secret key sk_{ID_i}.
- $\mathcal{IB}.Encrypt(pp, m, S)$: by taking the public parameters pp, a set of identities $S = \{ID_1, \ldots, ID_s\}$ and a message m as input, the algorithm generates a pair (Hdr, K) by utilizing S and pp, where Hdr denotes a header and K denotes the encryption key. The algorithm broadcasts (Hdr, S, C_M).
- $\mathcal{IB}.Decrypt(S, ID_i, sk_{ID_i}, Hdr, pp)$: by taking the identity set $S = \{ID_1, \ldots, ID_s\}$, an user's identity ID_i with secret key sk_{ID_i}, the header Hdr and the public parameters pp as input, the algorithm outputs the encryption key K. The ciphertext C_M can be decrypted by K.

D. Randomized Convergent Encryption (RCE)

Convergent encryption (CE) is introduced by Douceur et al. [12]. The core idea of CE is to generate the same ciphertext of the same file. It has been widely used in the data deduplication area. To resist brute force attack attack, we adopt randomized convergent encryption (RCE) scheme. The RCE has been proved secure in tag consistency [11]. A REC scheme consists of three algorithms. For the convenience, the notion $\mathcal{RCE}$ represents RCE.

- $\mathcal{RCE}.KeyGen(M, \lambda) \rightarrow K_c$: It takes the data M as input, and outputs a convergent key K_c.
- $\mathcal{RCE}.Encrypt(k_l, K_c, M) \rightarrow \{e_1, e_2\}$: It takes the convergent key K_c, a random key k_l and the data M as input, and outputs a ciphertext e_1 and an encrypted key e_2. M is encrypted by a symmetric encryption(e.g., AES 128) as $e_1 = Enc(k_l, M)$, and k_l is encrypted by K_c as $e_2 = k_l \oplus K_c$.
- $\mathcal{RCE}.Decrypt(K_c, e_1, e_2) \rightarrow M$: It takes the convergent key K_c, the ciphertext e_1 as well as the encrypted key e_2 as input and outputs the message M.

E. Proof of Ownership

Proof of ownership (PoW) requires a data owner to prove the possession of a file without uploading the file. The notion of PoW was first proposed by Halevi et al. [39], and they utilize merkle hash tree to achieve PoW. Deswarte et al. [40] proposed a "challenge-response" model to check the integrity of the outsourced data withour downloading the file. Up to now, the "challenge-response" model has been widely used in data deduplication and data integrity auditing, it usually has three algorithms:

- $Challenge$: The verifier makes a challenge for the verifier, and sends the challenge information to the verifier.
- $Proof$: The prover generates a response based on the challenge information.
- $Verify$: The verifier verifies the response and outputs the checking result.

IV. Problem Formulation

A. Design Goals

- *Data confidentiality:* to make sure that the CSP cannot deduce the content of data.
- *Key deduplication:* to reduce the storage overhead of the user side, the number of stored file keys on the user side is irrelevant to the number of files.
- *Tag deduplication:* to reduce the storage overhead of the cloud side, the number of auditing tags is irrelevant to the number of users.
- *Decentralization:* to reduce the power of a single TPA, the auditing process does not rely on a centralized TPA.
- *Batch Auditing:* to improve the auditing efficiency, the scheme supports one user with multiple files auditing and many users with the same file auditing at the same time.

B. Threat Model

In the threat model, we consider threat from semi-trusted CSP side and misbehaved auditor.

Semi-trusted CSP: The data loss may happen due to hardware failures. The CSP may always claim that the data are well

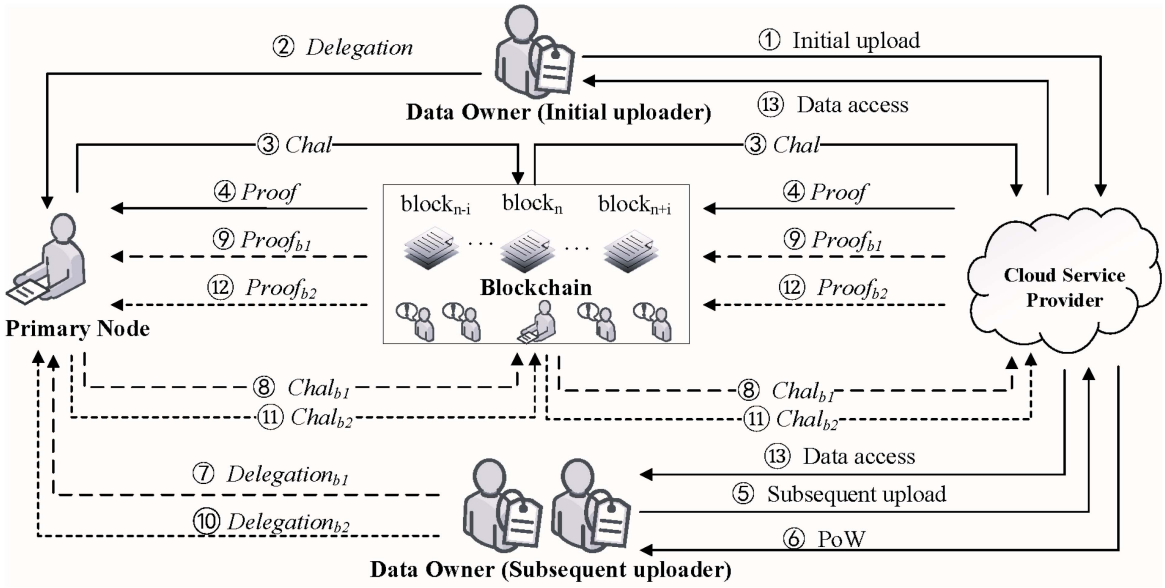


Fig. 1. System model.

retained for its commercial reputation when the data are lost. The CSP is honest but curious about the data content.

Misbehaved auditor: Less than a third of the auditors may be corrupted. The auditor may hide the result of data corruption and publish the wrong result to attempt to reach a consensus.

C. System Model

As is shown in Fig. 1, the system model includes the following three entities:

- *Data Owner (DO):* The DO outsources its file F to the cloud. The cloud storage provider checks the file state, if the file has not been uploaded yet, the DO should upload the file F and the auditing information. If the file exists, the DO should proof of ownership of F, and upload some auxiliary information. As part of the blockchain, the DO joins the blockchain and becomes a node in it.
- *Cloud Service Provider (CSP):* The CSP provides significant storage services for the DO. As part of blockchain, the CSP joins the blockchain and becomes a node in it as well.
- *Blockchain:* As a public distributed ledger, blockchain is maintained by various nodes(i.e., data owner node, cloud storage provider node and some other nodes), in which the primary node processes the transactions or record it.

The system runs as follows: 1) The DO outsources its data into the cloud, and then 2) makes a delegation $delegation$ for the primary node. The CSP first checks the data duplication.

- If it does not exist, the DO is a initial uploader. 3) The primary node makes a challenge $Chal$ for the CSP according to blockchain. 4) After receiving the challenge information, the CSP generates the corresponding proof information $Proof$ and returns it back to the blockchain. The whole network node cooperates with other nodes and verify the validation of the proof information. If the validation passes, the node confirms it and broadcasts it to the other nodes. Otherwise, the node refuses to confirm it. After receiving at least 2/3 node's confirmation, the primary node shares the auditing result on the blockchain.
- If it exists, the DO is a subsequent uploader. 6) The DO makes a proof of ownership PoW of the file. If the file does not exist, the DO should act as the initial uploader and upload the file and some auxiliary information. 7) When the DO has many files to auditing, the DO makes a delegation of multiple files $Delegation_{b_1}$ checking for for the primary node. 8) The primary node makes a batch challenge $Chal_{b_1}$ for the CSP according to blockchain. 9) After receiving the batch challenge information, the CSP generates the corresponding proof information $Proof_{b_1}$ and returns it back to the blockchain. The whole network node cooperates with other nodes and verify the validation of the proof information. If the validation passes, the node confirms it and broadcasts it to the other nodes. Otherwise, the node refuses to confirm it. After receiving at least 2/3 node's confirmation, the primary node shares the auditing result on the blockchain. 10) When the primary node receives many auditing tasks for the same file auditing $Delegation_{b_2}$. 11) the primary node makes a challenge $Chal_{b_2}$ for the CSP according to blockchain. 12) After receiving the batch challenge information, the CSP generates the corresponding proof information $Proof_{b_2}$ and returns it back to the blockchain. The whole network node cooperates with other nodes and verify the validation of the proof information. If the validation passes, the node confirms it and broadcasts it to the other nodes. Otherwise, the node refuses to confirm it. After receiving at least 2/3 node's confirmation, the primary node shares the auditing result on the blockchain.

The DO can access its data according to download request. The CSP returns the corresponding data information. The DO decrypts it and verifies the correctness of the data. The DO accepts the data only the data pass the verification.

The proposed scheme includes System Setup, Data Upload and Deduplication, Data Auditing, Batch Auditing and Data Access five protocols.

1) *System Setup:* This protocol includes System Parameters Generation (SPG) and DO Parameters Generation (DOPG).
   - *System Parameters Generation (SPG):* by input a security parameter λ, this protocol outputs the public parameters pp.
   - *DO Parameters Generation (DOPG):* This protocol outputs the private key and the public key pair (sk, pk) for users.
2) *Data Upload and Deduplication:* This protocol includes initial upload phase and subsequent upload phase.
   For the initial upload phase, it includes the following algorithms.
   - $\mathcal{RCE}.KeyGen, \mathcal{RCE}.Encrypt$: by taking the file F as input, the algorithms output the encrypted file encryption key K_l.
   - *TagGen, AuthGen:* by taking the file encryption key k_l as input, the algorithms output the authenticator $\Omega = \{\sigma_i\}_{1 \leqslant i \leqslant n}$, file public key pk_F and tag τ.
   - $\mathcal{IB}.Encrypt$*:* by taking the public parameters pp and the identity set S as input, the algorithms output the IBBE header Hdr.

   For the subsequent upload phase, it includes the following algorithms.
   - *PoW.Chal:* by taking the public parameters as input, the algorithm outputs the challenge information.
   - *PoW.Proof:* by taking the challenge index set I as input, the algorithm outputs the corresponding proof information.
   - *PoW.Verify:*by taking the proof information as input, the algorithm outputs the checking result.
3) *Data Auditing:* The protocol to achieve the data integrity auditing interacts as follows.
   The interactive protocol to realize the data integrity auditing is as follows.
   - *Audit.Chal:* by taking the public parameters as input, the algorithm outputs the challenge information $Chal$.
   - *Audit.Proof:* by taking the index of the challenged blocks I as input, the algorithm outputs the corresponding proof information $Proof$.
   - *Audit.Verify:* by taking the proof information $Proof$ as input, the algorithm outputs the verification result $true/false$.
4) *Batch Auditing:* To achieve batch auditing, we consider the following two situations:

a) One user with multiple files
   - *B1-Audit.Chal:* by taking the public parameters as input, the algorithm outputs the challenge information $Chal_{b1}$.
   - *B1-Audit.Proof:* by taking the challenge blocks set I as input, the algorithm outputs the corresponding proof information $Proof_b$.
   - *B1-Audit.Verify:* by taking the proof information $Proof_{b1}$ as input, the algorithm outputs the verification result $true/false$.

b) Many users with the same file
   - *B2-Audit.Chal:* by taking the public parameters as input, the algorithm outputs the challenge information $Chal_{b2}$.
   - *B2-Audit.Proof:* by taking the index of the challenged blocks I as input, the algorithm outputs the corresponding proof information $Proof_{b2}$.
   - *B2-Audit.Verify:* It takes the proof information $Proof_{b2}$ as input, and outputs the verification result $true/false$.

5) *Data Access:* To obtain the plaintext file F, the DO executes the following algorithms.
   - $\mathcal{IB}.Decrypt$: by taking $(S_F, ID_t, sk_{ID_t}, Hdr_F, pp)$ as input, the algorithm outputs the IBBE key K_{IBF}.
   - **Dec**$_{SE}$: by taking (K_{IBF}, d_F) as input, the algorithm outpust the convergent key K_c. According to K_c, it can recover the file encryption key k_l.
   - **Dec**: by taking the file encryption key k_l as input, the algorithm outputs the plaintext file F.

D. Security Model

The security model is based Shacham and Waters's work [41]. Suppose the adversary $\mathcal{A}$ is considered as untrust CSP. The $\mathcal{A}$ tries to forge a valid proof which can pass the verification. The challenger $\mathcal{C}$ is considered as the DO. The security model is defined as follows.

- *Initialization:* The $\mathcal{C}$ executes System Setup algorithm, including system parameters generation and DO parameters generation, to obtain public parameters. The $\mathcal{C}$ then sends the public parameters to $\mathcal{A}$.
- *Query:* The $\mathcal{A}$ makes the following queries and $\mathcal{C}$ answers the queries respectively.

1) $Hash\,Query$: $\mathcal{A}$ adaptively queries the hash values to $\mathcal{C}$. $\mathcal{C}$ responds the information to $\mathcal{A}$.
2) $Private\,Key\,Query$: $\mathcal{A}$ sends the file F to query its private key. $\mathcal{C}$ runs the $Authenticator\ generation$ algorithm to generate the private key of F and sends it to $\mathcal{A}$.
3) $Public\,Key\,Query$: $\mathcal{A}$ sends (ID_t, F) to query the file public key and the ID_t's file public key. $\mathcal{C}$ runs the $Authenticator\ generation$ algorithm to obtain the public key and sends them to $\mathcal{A}$.
4) $Authenticator\,Query$: $\mathcal{A}$ adaptively makes query for any block's corresponding authenticator from (ID_t, F). $\mathcal{C}$ runs the $Authenticator\ generation$ algorithm to obtain the authenticator and responds it to $\mathcal{A}$.

- *Challenge:* $\mathcal{C}$ runs the $Audit.Chal$ algorithm to generate the challenge information of (ID_t, F). The ID_t's private key of F should not be queried before. The authenticator of challenged blocks should not be queried before as well.
- *Output:* $\mathcal{A}$ outputs a forged proof based the challenge information.
- *ProofVerify:* $\mathcal{A}$ executes $Audit.Verify$ algorithm to verify the validation of the proof. If the forged proof can pass the verification, $\mathcal{A}$ wins this game.

Definition 3: The proposed scheme can resist untrusted CSP, if there exists any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the game is negligible.

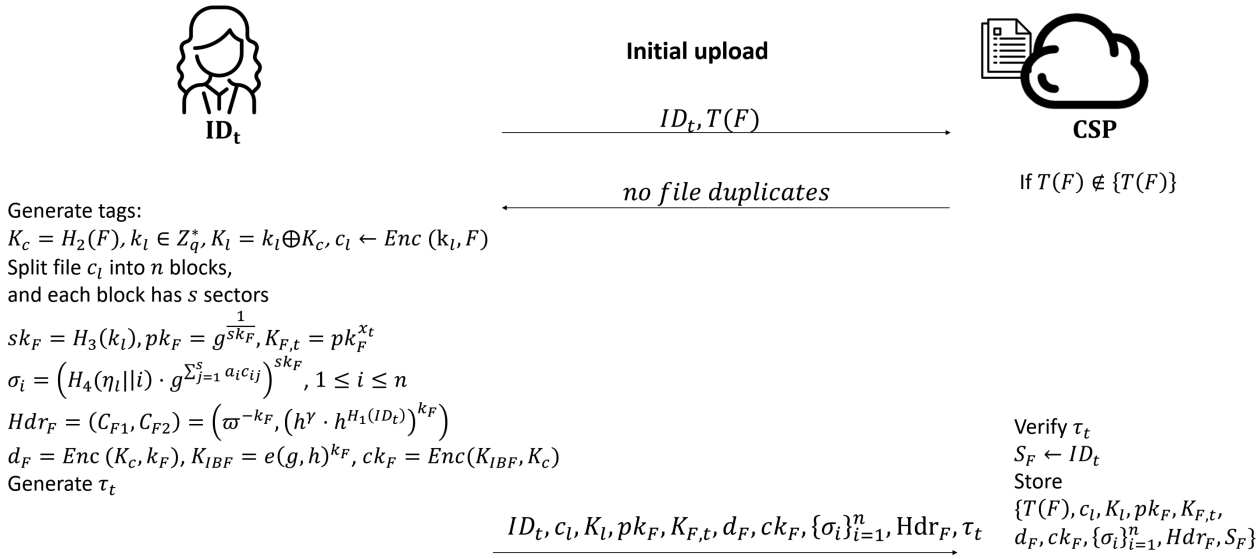


Fig. 2. Initial upload.

V. The Proposed Scheme

A. System Setup

This protocol contains two algorithms: **System Parameters Generation (SPG)** and **DO Parameters Generation (DOPG)**. Let $\mathbb{G}$ and $\mathbb{G}_T$ denote two multiplicative cyclic groups of order p. Let e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ denote a bilinear map. The hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is utilized to generate a tag and participates in the IBBE header generation. The hash function $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is utilized to generate the convergent key. The hash function $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is utilized to generate the private key. The AES-128 algorithm is utilized as encryption/decryption. $c \leftarrow Enc(k, m)$ denotes the encryption of message m using key k. $m \leftarrow Dec(k, c)$ denotes the decryption of ciphertext c using key k.

1) *SPG:* The *Key Generation Center* (KGC) adopts $\mathcal{IB}.setup$ to generate the master key, some system public parameters and DO's secret keys. The secret keys are used in IBBE algorithm. The KGC randomly picks some parameters: a large integer m, a security parameter λ, a large prime m where $|p|$ equals λ, a secret value $\gamma \in \mathbb{Z}_q^*$ and two generators $g, h \in \mathbb{G}$. After that, the KGC sets $\varpi = g^{\gamma}$ and $v = e(g, h)$. The master secret key in this system is $msk = (g, \gamma)$. When a new $DO\ ID_t$ enrolls in the system, the KGC generates $sk_{ID_t} = g^{\frac{1}{\gamma + H_1(ID_t)}}$, and transmits it to the DO ID_t. For the benefit of data deduplication, the KGC randomly selects $n, s \in \mathbb{Z}_q^*$, where n denotes the block number and s denotes the sector number. The system public parameters are $pp = (\varpi, v, H, h^{\gamma}, \ldots, h^{\gamma^m}, n, s)$.
2) *DOPG:* the ID_t generates his/her public-private keys to be used in the integrity auditing process. The ID_t randomly picks an integer $x_t \in \mathbb{Z}_q^*$ and then generates $pk_t = g^{x_t}$. The ID_t randomly chooses s elements $\{a_k \in \mathbb{Z}_q^*\}_{1 \leqslant k \leqslant s}$ and computes $\{u_k = g^{a_k}\}_{1 \leqslant k \leqslant s}$ as public values. The ID_t then randomly generates a signing public/private key pair (spk_t, ssk_t). Finally, the ID_t sets the public key as $pk = \{spk_t, pk_t\}$ and the private key as $sk = (x_t, ssk_t)$.

B. Data Upload and Deduplication

This protocol contains **Initial Upload** phase and **Subsequent Upload** phase. Suppose that the ID_t wants to send the original data F to the CSP. The ID_t retrieves the convergent key $K_c \leftarrow H_2(F)$, which will be used to encrypt the file encryption key. The ID_t computes a tag $T(F) = H_1(Enc(K_c, F))$ for F and sends a request $Upload = \langle ID_t, T(F) \rangle$ to the CSP. When receiving the $Upload$ request, the CSP stores ID_t and checks whether $T(F)$ exists. If not, it is an initial upload. If so, it is a subsequent upload.

Initial Upload: For a unique file F, the specific interaction flow is shown in Fig. 2. ID_t runs $\mathcal{RCE}.KeyGen$, $\mathcal{RCE}.Encrypt$, TagGen, AuthGen, $\mathcal{IB}.Encrypt$ to generate block ciphertext set c_l, block tag set $\Omega_l = \{\sigma_i\}_{1 \leqslant i \leqslant n}$, and IBBE header as follows.

1) *Data encryption:* The ID_t randomly chooses a value $k_l \in \mathbb{Z}_q^*$ as the file encryption key, which will be used to encrypt the data. After that, the ID_t generates the ciphertext of F as $c_l \leftarrow Enc(k_l, F)$, and the encrypted file encryption key as $K_l \leftarrow k_l \oplus K_c$. Then, the ID_t splits the ciphertext c_l into $c_l = \{c_{ij}\}_{(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s)}$.
2) *Authenticator generation:* The ID_t first generates the private key of file F as $sk_F = H_3(k_l)$(This key will be used for generating the authenticator), the corresponding public key as $pk_F = g^{\frac{1}{sk_F}}$, and the public auditing key of ID_t as $K_{F,t} = pk_F^{x_t}$. Suppose the data identity be $\eta_l = H_1(K_l)$. Then, the ID_t computes the authenticator

$$\sigma_i = \left(H_4(\eta_l \| i) \cdot g^{\sum_{j=1}^{s} a_j c_{ij}}\right)^{sk_F}, 1 \leqslant i \leqslant n.$$

Let $\Omega_l = \{\sigma_i\}_{1 \leqslant i \leqslant n}$ be the authenticators set.

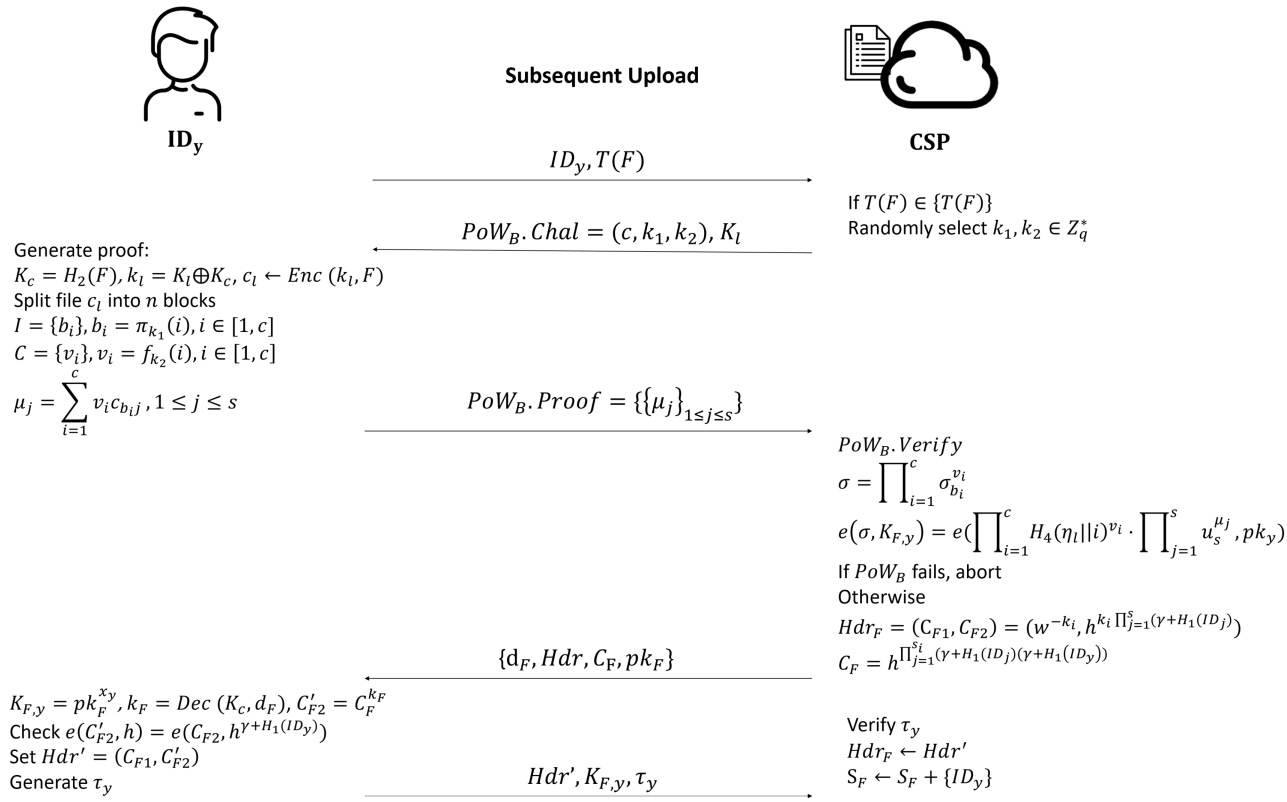


Fig. 3. Subsequent upload.

3) *IBBE encryption:* The ID_t randomly selects $k_F \in \mathbb{Z}_q^*$ and generates IBBE header

$$Hdr_i = (C_{F1}, C_{F2}) = \left(\varpi^{-k_F}, \left(h^{\gamma} \cdot h^{H_1(ID_t)}\right)^{k_F}\right).$$

Then the ID_t utilizes symmetric encryption to encrypt k_F as $d_F = Enc(K_c, k_F)$. It then utilizes IBBE's negotiate key to encrypt K_c as $ck_F = Enc(K_{IBF}, K_c)$.

Let the data tag be $\tau_t \leftarrow \tau_0 \| SSig_{ssk_t}(\tau_0)$, where $\tau_0 \leftarrow H(ID_t \| \eta_l \| n \| u_1 \| \cdots \| u_s \| K_l \| pk_F \| K_{F,t} \| Hdr_F)$ and $SSig_{ssk_t}(\tau_0)$ denotes a signature on τ_0 under the private key ssk_t. Finally, it uploads $(ID_t, c_l, K_l, pk_F, K_{F,t}, \{\sigma_i\}_{i=1}^n, Hdr_F, \tau_t)$ to the CSP. When receiving this information, the CSP verifies τ_t. Then, the CSP verifies the tag consistency by $e(\prod_{i=1}^n \sigma_i, K_{F,t}) = e(\prod_{i=1}^n H_4(\eta_l \| i) \cdot \prod_{i=1}^n \prod_{j=1}^s u_j^{c_{ij}}, pk_t)$. If all of these information pass the verification, the CSP adds ID_t into S_F and stores this information.

Subsequent Upload: For a subsequent uploader ID_y, the specific interaction flow is shown in Fig. 3. The CSP randomly selects $k_1, k_2 \in \mathbb{Z}_q^*$, $c \in [1, n]$ as the challenge information, and sends **PoW.Chal** $= (c, k_1, k_2)$ and K_l to ID_y. Upon receiving this information, the ID_y generates the proof as follows: the ID_y first retrieves convergent key $K_c = H_2(F)$ and decrypts the file encryption key by $k_l = K_l \oplus K_c$. The ID_y then encrypts file F by $c_l \leftarrow E(k_l, F)$, and splits c_l into n blocks. According to the challenge information, the ID_y generates the challenge content $I = \{b_i\}_{i=1}^c, b_i = \pi_{k_1}(i)$ for $1 \leqslant i \leqslant c$ and $C = \{v_i\}_{i=1}^c$, $v_i = f_{k_2}(i)$ for $1 \leqslant i \leqslant c$. The ID_y then generates the proof information

$$\mu_j = \sum_{i=1}^{c} v_i \cdot c_{b_i j}, 1 \leqslant j \leqslant s.$$

The ID_y responds the proof information **PoW.Proof** $= \{\{\mu_j\}_{1 \leqslant j \leqslant s}\}$. Upon receiving the proof information, the CSP verifies

$$\begin{cases} \sigma = \prod_{i=1}^c \sigma_{b_i}^{v_i} \\ e(\sigma, K_{F,y}) = e(\prod_{i=1}^c H_4(\eta_l \| i)^{v_i} \cdot \prod_{j=1}^s u_s^{\mu_j}, pk_y). \end{cases}$$

If **PoW.Verify** fails, the CSP aborts. Otherwise, the CSP computes

$$C_F = h^{\prod_{j=1}^{s_i}(\gamma + H_1(ID_j))(\gamma + H_1(ID_y))}. \quad (1)$$

The CSP can not obtain C_F directly, but it can obtain it as follows:

$$C_F = h^{\prod_{j=1}^{s_i}(\gamma + H_1(ID_j))(\gamma + H_1(ID))}$$

$$= h^{(\gamma + H_1(ID_1)) \cdot (\gamma + H_1(ID_2)) \cdots (\gamma + H_1(ID_{s_i})) \cdot (\gamma + H_1(ID_y))}$$

$$\xrightarrow{\quad let\ ID_{s_i+1} = ID_y \quad}$$

$$h^{\gamma^{s_i+1} + \gamma^{s_i} \sum_{j=1}^{s_i+1} H_1(ID_j) + \gamma^{s_i - 1} \sum_{j=t=1, j \neq t}^{s_i+1} H_1(ID_j) H_1(ID_t)}$$

$$\cdot h^{\cdots + \prod_{j=1}^{s_i+1} H_1(ID_j)}$$

$$= h^{\gamma^{s_i+1}} \cdot h^{\gamma^{s_i} \sum_{j=1}^{s_i+1} H_1(ID_j)}.$$

$$h^{\gamma^{s_i-1} \sum_{j=t=1, j\neq t}^{s_i+1} H_1(ID_j)H_1(ID_t)} \cdots h^{\prod_{j=1}^{s_i+1} H_1(ID_j)}.$$

The CSP then returns $\{d_F, Hdr, C_F, pk_F\}$ to the ID_y. Upon receiving this information, the ID_y generates the public auditing key $K_{F,y} = pk_F^{x_y}$. The ID_y decrypts k_F as $k_F = Dec(K_c, d_F)$ and uses it to generate $C'_{F2} = C_F^{k_F}$. The ID_y checks the validity of C'_{F2} by

$$e(C'_{F2}, h) = e(C_{F2}, h^{\gamma + H_1(ID)}).$$

If the verification passes, the ID_y sets $Hdr' = (C_{F1}, C'_{F2})$. Then, the ID_y sets a verification tag $\tau_y \leftarrow \tau_1 \| SSig_{ssk_y}(\tau_1)$, where $\tau_1 \leftarrow H(Hdr' \| K_{F,y})$. Finally, the ID_y responses with $(Hdr', K_{F,y}, \tau_y)$ to the CSP. Upon receiving this information, the CSP verifies τ_y. If it passes the verification, the CSP sets $Hdr_F \leftarrow Hdr'$ and adds ID_y into S_F.

The negotiated secret key K_{IBF} among S_F can be recovered as follows:

Let S'_F be $S_F - ID_y$ and s'_F be $|S'_F|$. Let $p_s(\gamma) = \frac{1}{\gamma}(\prod_{j=1}^{s'_F}(\gamma + H(ID_j)) - \prod_{j=1}^{s'_F} H_1(ID_j))$.

As for $h^{p_s(\gamma)} = h^{\frac{1}{\gamma}(\prod_{j=1}^{s'_F}(\gamma+H_1(ID_j)) - \prod_{j=1}^{s'_F} H_1(ID_j))}$, its computation method is the same as C_F. We omit it here.

The IBBE key can be recovered by ID_y as follows:

$$\left[e\left(C_{F1}, h^{p_s(\gamma)}\right) \cdot e\left(sk_{ID_y}, C_{F2}\right)\right]^{\frac{1}{\prod_{j=1}^{s'_F} H_1(ID_j)}}$$

$$= \left[e\left(\varpi^{-k_F}, h^{\frac{1}{\gamma}\left(\prod_{j=1}^{s'_F}(\gamma+H_1(ID_j)) - \prod_{j=1}^{s'_F} H_1(ID_j)\right)}\right)\right.$$

$$\left. e\left(g^{\frac{1}{\gamma+H_1(ID_y)}}, h^{k_F \prod_{j=1}^{s_i}(\gamma+H(ID_j))}\right)\right]^{\frac{1}{\prod_{j=1}^{s'_F} H_1(ID_j)}}$$

$$= \left[e(g^{-\gamma k_F}, h^{\frac{1}{\gamma}(\prod_{j=1}^{s'_F}(\gamma+H_1(ID_j)) - \prod_{j=1}^{s'_F} H_1(ID_j))}) \cdot \right.$$

$$\left. e(g,h)^{\frac{1}{\gamma+H_1(ID_y)} \cdot k_i \prod_{j=1}^{s_i}(\gamma+H_1(ID_j))}\right]^{\frac{1}{\prod_{j=1}^{s'_F} H_1(ID_j)}}$$

$$= \left[e(g,h)^{k_F \prod_{j=1}^{s'_F} H_1(ID_j)}\right]^{\frac{1}{\prod_{j=1}^{s'_F} H_1(ID_j)}}$$

$$= e(g,h)^{k_F} = v^{k_F} = K_{IBF}.$$

C. Data Auditing

The *Auditing* protocol is executed among DO, CSP and blockchain. Fig. 4 shows the auditing process based PBFT.

1) The DO first makes a delegation $Delegation = \langle ID_t, T(F)\rangle$ for the primary node. The primary node runs the **Audit.Chal** algorithm to generate a challenge $Chal = (c, k_1, k_2)$, where $c \in [1, n]$ and $k_1, k_2 \in \mathbb{Z}_q^*$. Then the primary node records the challenge information to the blockchain.

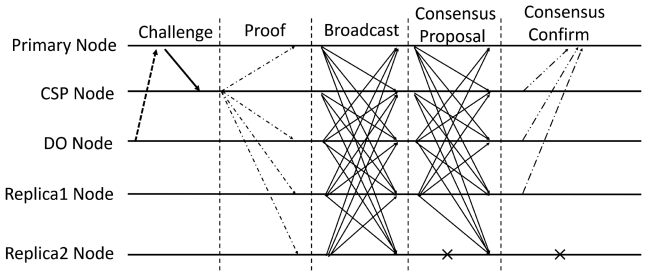


Fig. 4. Flow of challenge and proof verification phase based PBFT consensus mechanism.

2) After receiving $chal$ from the blockchain, the CSP performs the **Audit.Proof** algorithm to obtain the corresponding proof. Specifically, the CSP first generates the index set $I = \{b_i\}_{i=1}^c, b_i = \pi_{k_1}(i)$ for $1 \leqslant i \leqslant c$ and the coefficient set $C = \{v_i\}_{i=1}^c, v_i = f_{k_2}(i)$ for $1 \leqslant i \leqslant c$. The CSP then aggregates the authenticators and the linearly combines the metadata sectors as

$$\begin{cases} \sigma = \prod_{i=1}^c \sigma_{b_i}^{v_i} \\ \mu_j = \sum_{i=1}^c v_i \cdot c_{b_i j}, \quad 1 \leqslant j \leqslant s. \end{cases}$$

Then, the CSP publishes the aggregated integrity proof $Proof = \{\sigma, \{\mu_j\}_{j=1}^s\}$ into the transaction pool.

3) When receiving the integrity proof $Proof$ from the transaction pool, each node involved in maintaining the blockchain performs the **Audit.Verify** algorithm to verify the validity of $Proof$ as follows.

$$e(\sigma, K_{F,t}) = e\left(\left(\prod_{i=1}^c H_4(\eta_l \| i)^{v_i} \cdot \prod_{j=1}^s u_s^{\mu_j}, pk_t\right)\right. . \tag{2}$$

If the equation holds, the node confirms it and broadcasts it to other nodes. Otherwise, the node refuses to confirm it. After receiving at least $2/3$ node's confirmation, the primary node shares the auditing result $result = \langle Proof, true/false\rangle$ on the blockchain.

D. Batch Auditing

There are two conditions to consider: a data owner wants to check multiple files or there are many users who want to check the integrity of the same file.

1) One User With Multiple Files Auditing: Suppose a user ID_y wants to audit the integrity of multiple files $\{F_1, F_2, \ldots, F_d\}$ at once. The batch auditing works as follows.

1) The ID_y makes a delegation $Delegation_{b1} = \langle ID, T(F_1), \cdots T(F_d)\rangle$ for the current primary node. The primary node first runs the **B1-Audit.Chal** algorithm to retrieve the challenge information $Chal_{b1} = (c, k_1, k_2, k_3)$, where $c \in [1, n]$ and $k_1, k_2, k_3 \in \mathbb{Z}_q^*$. Then the primary node records the challenge information $Chal_{b1}$ to the blockchain.
2) After receiving $Chal_{b1}$ from the blockchain, the CSP executes the **B1-Audit.Proof** algorithm to obtain the corresponding proof. Specifically, the CSP first calculates the index set $I_{b_1} = \{b_i\}_{i=1}^c$, $b_i = \pi_{k_1}(i)$ for $1 \leqslant i \leqslant c$ and

the coefficient set $C_{b_1} = \{v_i\}_{i=1}^{c}$, $X_{b_1} = \{x_i\}_{i=1}^{c}$, $v_i = f_{k_2}(i), x_i = f_{k_3}(i)$ for $1 \leqslant i \leqslant c$. The CSP then generates the aggregated authenticator and the linear combination of the metadata sectors as

$$\begin{cases} \sigma_{b1} = \prod_{k=1}^{d} \prod_{i=1}^{c} \sigma_{b_i k}^{x_k \cdot v_i} \\ \mu_{j1} = \sum_{k=1}^{d} \sum_{i=1}^{c} x_k \cdot v_i \cdot c_{b_i j}, \quad 1 \leqslant j \leqslant s. \end{cases}$$

Finally, the CSP publishes the aggregated integrity proof$Proof_{b1} = \{\sigma_{b1}, \{\mu_{j1}\}_{j=1}^{s}\}$ into the transaction pool.

3) When receiving the integrity proof $Proof_{b1}$ from the transaction pool, each node involved in maintaining the blockchain executes the **B1-Audit.Verify** algorithm to verify the correctness of $Proof_{b1}$ as follows.

$$e\left(\sigma_{b1}, \prod_{k=1}^{d} K_{F_k,y}\right) = e\Bigg(\left(\prod_{k=1}^{d}\prod_{i=1}^{c} H_4\left(\eta_k \| i\right)^{x_k \cdot v_i} \cdot \prod_{j=1}^{s} u_s^{\mu_{j1}}, pk_y^d\right)\Bigg). \tag{3}$$

When the equation holds, the node confirms it and broadcasts it to other nodes. Otherwise, the node refuses to confirm it. After receiving at least $2/3$ node's confirmation, the primary node shares the auditing result $result = \langle Proof_{b1}, true/false\rangle$ on the blockchain.

2) Many Users With the Same File Auditing: Assume there are many requests on the same file F_o auditing from different users in the transaction pool.

1) Assume that the primary node receives the delegation $Delegation_{b2} = \langle ID_1, \ldots, ID_d, T(F_o)\rangle$. The primary node runs **B2-Audit.Chal** algorithm to obtain the challenge information $Chal_{b2} = (c, k_1, k_2)$, where $c \in [1, n]$ and $k_1, k_2 \in \mathbb{Z}_q^*$. Then the primary node records the challenge information $Chal_{b_2}$ to the blockchain.
2) After receiving $Chal_{b2}$ from the blockchain, the CSP performs the **B2-Audit.Proof** algorithm to obtain the corresponding proof. The CSP first calculates the index set $I_{b_2} = \{b_i\}$, $b_i = \pi_{k_1}(i)$ for $1 \leqslant i \leqslant c$ and the coefficient set $C_{b_2} = \{v_i\}_{i=1}^{c}$, $v_i = f_{k_2}(i)$ for $1 \leqslant i \leqslant c$. The CSP then generates the aggregated authenticator and the linear combination of the metadata sectors as

$$\begin{cases} \sigma_{b2} = \prod_{i=1}^{c} (\prod_{k=1}^{d} \sigma_{b_i k}^{v_i}) \\ K_{F,U} = \prod_{j=1}^{d} K_{F,j} \\ \mu_{j2} = \sum_{i=1}^{c} d \cdot v_i \cdot c_{b_i j}, \quad 1 \leqslant j \leqslant s. \end{cases}$$

Then, the CSP publishes the aggregated integrity proof$Proof_{b2} = \{\sigma_{b2}, \{\mu_{j2}\}_{j=1}^{s}\}$ into the transaction pool.

3) When receiving the integrity proof $Proof_{b2}$ from the transaction pool, each node involved in maintaining the blockchain performs the **B2-Audit.Verify** algorithm to verify the correctness of $Proof_{b2}$ as follows.

$$e(\sigma_{b2}, K_{F,U}) = e\left(\prod_{i=1}^{c} H_4(\eta_o \| i)^{dv_i} \cdot \prod_{j=1}^{s} u_s^{\mu_{j2}}, \prod_{i=1}^{d} pk_i\right). \tag{4}$$

If the equation holds, the node confirms it and broadcasts it to other nodes. Otherwise, the node refuses to confirm it. After receiving at least $2/3$ node's confirmation, the primary node shares the auditing result $result = \langle Proof_{b2}, true/false\rangle$ on the blockchain.

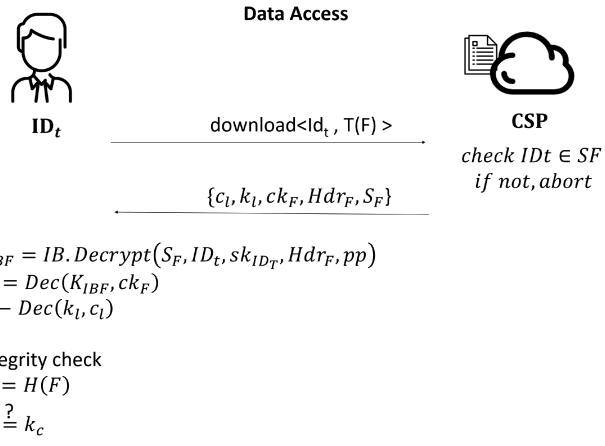


Fig. 5. Data access.

E. Data Access

When a user ID_t wants to download its file F, he/she sends a request $download = \langle ID_t, T(F)\rangle$ to the CSP, as shown in Fig. 5. When receiving the download request for the file F, the CSP first checks whether $ID_t \in S_F$. If not, the CSP aborts. Otherwise, the CSP returns $\{c_l, K_l, ck_F, Hdr_F, S_F\}$ to ID_t. Upon receiving this information, the ID_t first recovers the IBBE key K_{IBF} by $\mathcal{IB}.Decrypt(S_F, ID_t, sk_{ID_t}, Hdr_F, pp)$. The ID_t then decrypts the convergent key by using symmetric encryption $K_c = Dec(K_{IBF}, ck_F)$. Then, the ID_t can recover the file encryption key $k_l = K_l \oplus K_c$. By utilizing the file encryption key, the ID_t can decrypt the file $F \leftarrow Dec(k_l, c_l)$. To verify the correctness of the file, the ID_t can compute the convergent key $K_c' = H(F)$ and verify whether $K_c' = K_c$ holds. If so, the ID_t accepts the data. Otherwise, the ID_t aborts for acception.

VI. Security Analysis

Theorem 1: The auditing proof is correct if and only if the CSP maintains the data and the authenticators correctly.

Given an instance $\{g, g^a, h\}$, the CDH problem is to retrieve the element h^a. Given an instance $\{g, h = g^x\}$, the DL problem is to retrieve the value x. If the $\mathcal{A}$ has the ability to win the game with negligible probability, by using the capability of $\mathcal{A}$, a simulator $\mathcal{S}$ can be created to solve the CDH and DL problem with non-negligible probability.

Proof: Game 0: the $\mathcal{S}$ sets $sk_F = \alpha$ and $pk_F = g^{\frac{1}{\alpha}}$, $x_u = \beta$ and $pk_u = g^{\beta}$. The public auditing key of ID_u is $K_{F,u} = g^{\frac{\beta}{\alpha}}$. Then the $\mathcal{S}$ returns the public parameters to $\mathcal{A}$ and maintains

private keys secretly. In the query phase, the $\mathcal{A}$ can make $Hash\,Query$, $Private\,Key\,Query$, $Public\,Key\,Query$ and $Authenticator\,Query$ respectively. After that, the $\mathcal{C}$ makes a challenge to $\mathcal{A}$ and obtains the proof information $Proof = \{\sigma, \{\mu_j\}_{j=1}^{s}\}$. The $\mathcal{A}$ wins if the proof passes the verification.

Game 1: **Game1** is almost the same with **Game0**, except a little difference that the $\mathcal{S}$ maintains a list of τ_F. If the $\mathcal{A}$ responds a tag τ_F', which is valid but not in the list, the **Game1** aborts.

Analysis: If **Game1** aborts, it means the $\mathcal{A}$ sends a valid but unrecorded tag. If **Game1** does not abort, the tags are valid. That means $\mathcal{A}$ can forge a valid signature $SSig_{ssk}(\cdot)$, which disobey the unforgeability of signature used in this paper. Thus, the tag can not be tampered with by the $\mathcal{A}$.

Game2: **Game2** is almost the same with **Game1**, except a little difference that $\mathcal{S}$ maintains a list to keep the auditing proof. If the aggregated authenticator $\sigma^* = \prod_{i=1}^{c} \sigma_{b_i}^{v_i}$ does not exist in the list but still passes the verification, the **Game2** aborts.

Analysis: The valid proof is $Proof = \{\sigma, \{\mu_j\}_{j=1}^{s}\}$. According to the (5)

$$e\left(\sigma, K_{F,t}\right) = e\left(\left(\prod_{i=1}^{c} H_4\left(\eta_l \| i\right)^{v_i} \cdot \prod_{j=1}^{s} u_s^{\mu_j}, pk_t\right)\right). \quad (5)$$

If the $\mathcal{A}$ successfully forges the proof $Proof* = \{\sigma^*, \{\mu_j^*\}_{j=1}^{s}\}$ and passes the verification, which satisfies

$$e\left(\sigma^*, K_{F,t}\right) = e\left(\left(\prod_{i=1}^{c} H_4\left(\eta_l \| i\right)^{v_i} \cdot \prod_{j=1}^{s} u_s^{\mu_j^*}, pk_t\right)\right). \quad (6)$$

Since the $\mathcal{A}$ successfully forges the aggregated authenticator, there is at least one $\mu_j^* \neq \mu_j$. Let $\triangle\mu_j = \mu_j^* - \mu_j$. For each $1 \leqslant j \leqslant s$, the $\mathcal{S}$ randomly chooses $\alpha_j, \beta_j \in Z_q^*$, and sets $u_j = g^{\alpha_j} h^{\beta_j}$. For the hash query of $H_4(\eta_l \| i)$, the $\mathcal{S}$ randomly picks $r_i \in \mathbb{Z}_q^*$, and responses to the random oracle as $H_4(\eta_l \| i) = \frac{g^{r_i}}{\prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})^{c_{ij}}}$. For the authenticator query, the $\mathcal{S}$ responses

$$\begin{aligned}
\sigma_i &= (H_4(\eta_l \| i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}})^{\alpha} \\
&= \left(\frac{g^{r_i}}{\prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})} \cdot \prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})^{c_{ij}}\right)^{\alpha} \\
&= (g^{r_i})^{\alpha} \\
&= (g^{\alpha})^{r_i}.
\end{aligned}$$

By dividing the (2) by (6), we have

$$e\left(\frac{\sigma^*}{\sigma}, g^{\frac{\beta}{\alpha}}\right) = e\left(\prod_{j=1}^{s} u_j^{\triangle\mu_j}, g^{\beta}\right) = e\left(\prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})^{\triangle\mu_j}, g^{\beta}\right).$$

Furthermore, it can obtain

$$\left(\frac{\sigma^*}{\sigma}\right)^{\frac{\beta}{\alpha}} = \prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})^{\triangle\mu_j \beta} = g^{\sum_{j=1}^{s} \alpha_j \triangle\mu_j \beta} h^{\beta_j \triangle\mu_j \beta}.$$

Next, it can retrieve $\frac{\sigma^*}{\sigma} = g^{\sum_{j=1}^{s} \alpha_j \triangle\mu_j \alpha} \cdot h^{\sum_{j=1}^{s} \beta_j \triangle\mu_j \alpha}$. The $\mathcal{S}$ can retrieve $h^{\alpha} = \left[\left(\frac{\sigma^*}{\sigma} \cdot g^{-\sum_{j=1}^{s} \alpha_j \triangle\mu_j \alpha}\right)\right]^{\frac{1}{\sum_{j=1}^{s} \beta_j \triangle\mu_j}}$. As long as $\sum_{j=1}^{s} \beta_j \triangle \mu_j \neq 0$, the solution to the CDH problem is valid. Thus, the probability to solve the CDH problem is $1 - \frac{1}{q}$.

Game3: **Game3** is almost the same with **Game2**, except a little difference that the $\mathcal{S}$ matains a list of the auditing proof. If the aggregated data information $\mu_j = \sum_{j=1}^{s} v_i c_{b_i j}, 1 \leqslant j \leqslant s$ does not appear in the list but can pass the verification, **Game3** aborts.

Analysis: Since the forged proof passes the verification, it satisfies

$$e\left(\frac{\sigma^*}{\sigma}, g^{\frac{\beta}{\alpha}}\right) = e\left(\prod_{j=1}^{s} u_j^{\triangle\mu_j}, g^{\beta}\right) = e\left(\prod_{j=1}^{s}(g^{\alpha_j} h^{\beta_j})^{\triangle\mu_j}, g^{\beta}\right).$$

The $\mathcal{S}$ behaves almost the same in **Game2**, except the following:

For each $1 \leqslant j \leqslant s$, the $\mathcal{S}$ randomly selects $\alpha_j, \beta_j \in \mathbb{Z}_q^*$ and sets $\mu_j = g^{\alpha_j} h^{\beta_j}$. From **Game2**, we can know that $\sigma = \sigma^*$. Thus, it can obtain $\prod_{j=1}^{s} u_j^{\mu_j} = \prod_{j=1}^{s} u_j^{\mu_j^*}$. Define $\triangle\mu_j = \mu_j^* - \mu_j$. Therefore, the solution of DL problem is $h = g^{-\frac{\sum_{j=1}^{s} \alpha_j \triangle\mu_j}{\sum_{j=1}^{s} \beta_j \triangle\mu_j}}$. As long as $\sum_{j=1}^{s} \beta_j \triangle \mu_j \neq 0$, the probability to solve the DL problem is $1 - \frac{1}{q}$.

Theorem 2: The proposed scheme satisfies the property of tag deduplication.

Proof: In the proposed scheme, the DO generates the authenticators by using the file private key sk_F, which is $\sigma_i = (H_4(\eta_l \| i) \cdot g^{\sum_{j=1}^{s} a_j c_{ij}})^{sk_F}$. The file private key is retrieved from k_l and $k_l \leftarrow K_l \oplus K_c$. Since K_c is the convergent key, as long as the subsequent uploader possesses the same file, the subsequent uploader can recover the file's private key. Therefore, the proposed scheme guarantees the same file corresponding to the same authenticator. When different owners possess the same file, they possess the same authenticators as well. The CSP only requires to store a copy of the authenticators. The storage overhead greatly reduces from $\mathcal{O}(n)$ to $\mathcal{O}(1)$. Thus, the proposed scheme satisfies the property of tag deduplication.

Theorem 3: The proposed scheme satisfies the property of key deduplication.

Proof: In the proposed scheme, the DO only stores IBBE key sk_{ID_t}. According to the IBBE property, the DO can recover the negotiate key K_{IBF} and use it to decrypt the convergent key d_F. Thus, the ciphertext only needs to be stored once. When the identity exists in the indentity set S_F, the DO can decrypt it and then obtain the file encryption key. By utilizing the file encryption key, the DO can obtain the original file. While in previous scheme, one file corresponds to one convergent key. In order to retrieve the original file, the DO either stores the convergent key or distributes the convergent key to the key server. However, convergent key brings heavy storage cost. By distributing the convergent key to the key server, the DO needs to recover the convergent key every time when downloading the

TABLE II
COMPUTATION COSTS

Schemes	Initial Upload	Subsequent upload		Data auditing	
		PoW.proof	PoW.verify	Audit.proof	Audit.verify
Scheme [8]	$n \cdot (H + 2Exp_{\mathbb{G}} + Mul_{\mathbb{G}})$	$cMul_{\mathbb{Z}_q}$	$2Pair + (2c+1)Exp_{\mathbb{G}} + cH + Mul_{\mathbb{G}}$	$cExp_{\mathbb{G}} + cMul_{\mathbb{Z}_q}$	$2Pair + (c+1)Exp_{\mathbb{G}} + Mul_{\mathbb{G}} + cH$
Scheme [9]	$n \cdot (3H + 2Exp_{\mathbb{G}} + 1E + Mul_{\mathbb{G}})$	$n \cdot (3H + 2Exp_{\mathbb{G}} + 1E + Mul_{\mathbb{G}})$	Neg	$Pair + cMul_{\mathbb{Z}_q} + cExp_{\mathbb{G}} + Mul_{\mathbb{Z}_q}$	$2Pair + (c+3)Exp_{\mathbb{G}} + Mul_{\mathbb{G}} + cH$
Scheme [10]	$n \cdot (3H + 2Exp_{\mathbb{G}} + 1E + Mul_{\mathbb{G}})$	$c(2H + 1E + Mul_{\mathbb{Z}_q})$	$2Pair + +(2c+1)Exp_{\mathbb{G}} + cH + Mul_{\mathbb{G}}$	$cExp_{\mathbb{G}} + cMul_{\mathbb{G}}$	$(c+1)Exp_{\mathbb{G}} + cH + 2Pair + Mul_{\mathbb{G}}$
Our scheme	$w(2H + 2Exp_{\mathbb{G}} + 1E + sMul_{\mathbb{Z}_q} + Mul_{\mathbb{G}})$	$1E + cMul_{\mathbb{Z}_q}$	$2Pair + (c+s)Exp_{\mathbb{G}} + cH$	$cExp_{\mathbb{G}} + cMul_{\mathbb{Z}_q}$	$2Pair + (c+s)Exp_{\mathbb{G}} + cH$

H: hash operation. $Exp_{\mathbb{G}}$: exponentiation operation in $\mathbb{G}$. $Pair$: bilinear pairing operation.
$Mul_{\mathbb{G}}, Mul_{\mathbb{Z}_q}$: multiplication operation in $\mathbb{G}$ and $\mathbb{Z}_q$ respectively. $1E$: encryption operation in RCE.

TABLE III
COMMUNICATION COSTS

Schemes	Initial upload	Subsequent upload	Data auditing	Batch auditing
Scheme [8]	$n\|\mathbb{G}| + |F|$	$4|\mathbb{Z}_q|$	$4|\mathbb{Z}_q| + |\mathbb{G}|$	—
Scheme [9]	$(n+1)|\mathbb{G}| + |F|$	$(n+1)|\mathbb{G}| + |F|$	$(2c+1)|\mathbb{Z}_q| + 3|\mathbb{G}|$	$\mathcal{F}((2c+1)|\mathbb{Z}_q| + 3|\mathbb{G}|)$
Scheme [10]	$(n+1)|\mathbb{G}| + (n+1)|\mathbb{Z}_q| + |F|$	$4|\mathbb{Z}_q|$	$4|\mathbb{Z}_q| + |\mathbb{G}|$	$(c+5)|\mathbb{Z}_q| + |\mathbb{G}|$
Our scheme	$(w+6)|\mathbb{G}| + 3|\mathbb{Z}_q| + |F|$	$8|\mathbb{G}| + (s+5)|\mathbb{Z}_q|$	$4|\mathbb{Z}_q| + |\mathbb{G}|$	$(4+s)|\mathbb{Z}_q| + |\mathbb{G}|$

$|F|$: Length of the outsourced file F. $|\mathbb{G}|$: Length of the element in $\mathbb{G}$. $|\mathbb{Z}_q|$: Length of the element in $\mathbb{Z}_q$.

TABLE IV
STORAGE COSTS

Schemes	CSP	User	Blockchain
Scheme [8]	$\mathcal{U}n|\mathbb{G}| + |F|$	$|\mathbb{Z}_q|$	$5|\mathbb{Z}_q| + |\mathbb{G}|$
Scheme [9]	$\mathcal{U}(n+1)|\mathbb{G}| + |F|$	$(2 + \mathcal{F}n)|\mathbb{Z}_q|$	$3|\mathbb{G}| + |\mathbb{Z}_q|$
Scheme [10]	$(n+1)|\mathbb{G}| + (n+1)|\mathbb{Z}_q| + |F|$	$(\mathcal{F}+1)|\mathbb{Z}_q| + \mathcal{F}|\mathbb{G}|$	$5|\mathbb{Z}_q| + |\mathbb{G}|$
Our scheme	$(\mathcal{U}+w+4)|\mathbb{G}| + (\mathcal{U}+2)|Z_q| + |F|$	$4|\mathbb{Z}_q|$	$(3+s)|\mathbb{Z}_q| + |\mathbb{G}|$

$\mathcal{F}$: the number of files. $\mathcal{U}$: the number of users owns the same file F.

file. Thus, the proposed scheme satisfies the property of key deduplication.

Theorem 4: The proposed scheme satisfies the property of decentralization.

Proof: The proposed scheme utilize a consensus mechanism to replace a centralized TPA. If and only if more than $2/3$ nodes pass the verification and broadcast the auditing result to the other nodes, the result will be recorded into the blockchain. When an attacker wants to tamper with the data, it should control more than $2/3$ nodes. According to the security of blockchain, achiving this goal is difficult. As long as the corrupt nodes are less than $1/3$ of the whole network, the security can be guaranteed. This is also the underlying security of the blockchain. Thus, the proposed scheme satisfies the property of decentralization.

Theorem 5: The proposed scheme achieves data confidentiality.

Proof: In the proposed scheme, the DO generates the ciphertext of F as $c_l \leftarrow Enc(k_l, F)$, and splits the ciphertext c_l into $c_l = \{c_{ij}\}_{(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s)}$. The CSP cannot obtain the encryption key k_l. Thus, the CSP cannot deduce the plaintext. The data content cannot be leaked to the auditor. In the auditing process, the auditor obtains the proof information $Proof = \{\sigma, \{\mu_j\}_{j=1}^{s}\}$, where $\sigma = \prod_{i=1}^{c} \sigma_{b_i}^{v_i}$ and $\mu_j = \sum_{i=1}^{c} v_i \cdot c_{b_i j}, 1 \leqslant j \leqslant s$. From the information, the auditor cannot deduce the plaintext as well.

Theorem 6: The proposed scheme achieves data confidentiality.

Proof: In the proposed scheme, the DO generates the ciphertext of F as $c_l \leftarrow Enc(k_l, F)$, and splits the ciphertext c_l into $c_l = \{c_{ij}\}_{(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s)}$. The CSP cannot obtain the encryption key k_l. Thus, the CSP cannot deduce the plaintext. The data content cannot be leaked to the auditor. In the auditing process, the auditor obtains the proof information $Proof = \{\sigma, \{\mu_j\}_{j=1}^{s}\}$, where $\sigma = \prod_{i=1}^{c} \sigma_{b_i}^{v_i}$ and $\mu_j = \sum_{i=1}^{c} v_i \cdot c_{b_i j}, 1 \leqslant j \leqslant s$. From the information, the auditor cannot deduce the plaintext as well.

VII. THEORETICAL ANALYSIS AND PERFORMANCE ANALYSIS

A. Theoretical Analysis

In this section, we summarize the theoretical performance in computation cost, communication cost and storage cost. We compare our scheme with [8], [9], and [10].

Computation cost: Let H, $Pair$, $Exp_{\mathbb{G}}$, and $Mul_{\mathbb{G}}$ denote the operations of hash, pairing, exponentiation, and multiplication on group $\mathbb{G}$ respectively. Let $Mul_{\mathbb{Z}_q}$ denote multiplication on $\mathbb{Z}_q$. Let $1E$ denote RCE. The comparison result is shown in Table II. In the initial upload phase, the main computation cost is to generate the authenticators for all blocks. The computation cost in the scheme [8] is $n \cdot (H + 2Exp_{\mathbb{G}} + Mul_{\mathbb{G}})$. The computation cost in the scheme [9] and [10] is the same, which is $n \cdot (3H + 2Exp_{\mathbb{G}} + n \cdot (3H + 2Exp_{\mathbb{G}}$. Suppose the size of file F is $n|\mathbb{Z}_q|$. In our scheme, we split F into $w = \frac{n|\mathbb{Z}_q|}{s|\mathbb{Z}_q|}$ blocks. The

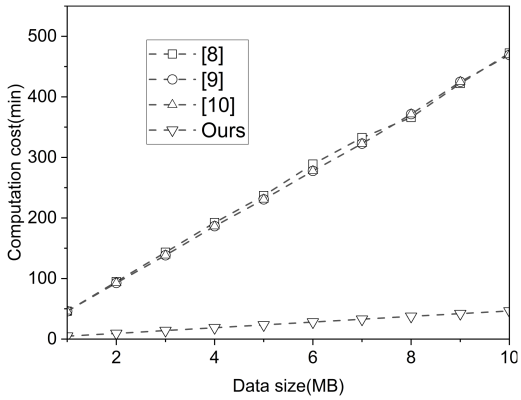


Fig. 6. Computation cost of initial upload.

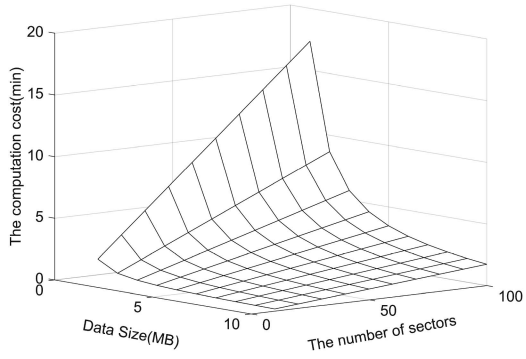
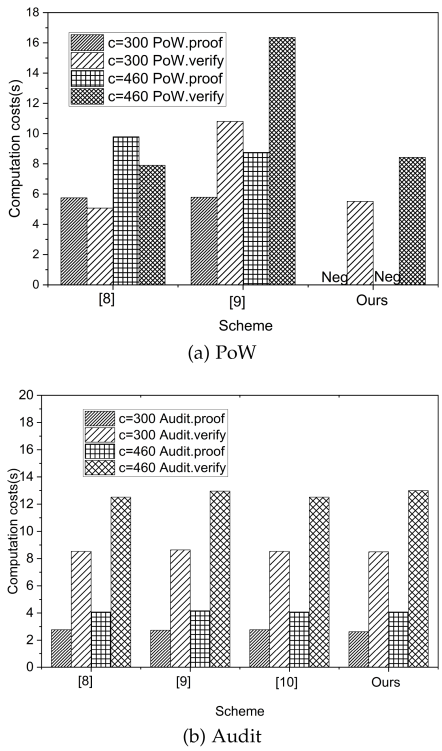


Fig. 7. Impact of data size and the number of sectors on our scheme.



(a) PoW

(b) Audit

Fig. 8. Computation costs of PoW and auditing when c=300 and 460.

computation cost $w(2H + 2Exp_{\mathbb{G}} + 1E + sMul_{\mathbb{Z}_q} + Mul_{\mathbb{G}})$. In general, $w \leqslant n$. Our scheme is efficient in authenticator generation. As for subsequent upload, a DO should prove the possession of the file. In scheme [8], the cost is $cMul_{\mathbb{Z}_q}$, which is simililar to us. Our scheme should generate the file encryption key k_l, so there's an extra encryption overhead. In scheme [9], the data deduplication is on the server side, so the computation cost is the same as the initial upload. In scheme [10], the computation cost is $c(2H + 1E) + Mul_{\mathbb{Z}_q}$. In the **Subsequent.Verify** and **Audit.Verify** phase, the computation cost is similar. Except that our computation cost is related to the number of sectors. In the **Audit.proof** phase, the computation cost is the same in the scheme [8], [10], and ours, which is $cExp_{\mathbb{G}} + cMul_{\mathbb{Z}_q}$. While in [9], it needs more than $Pair + Mul_{\mathbb{Z}_q}$ cost. But their scheme can achieve data privacy.

Communication cost: Let F denote the length of the outsourced file. Let $|\mathbb{G}|$ denote the length of the element in $\mathbb{G}$. Let $|\mathbb{Z}_q|$ denote the length of the element in $\mathbb{Z}_q$. The comparison result is shown in Table III. In the initial upload phase, other schemes need to upload n authenticators. While our scheme only needs w authenticators, which can reduce the storage cost. In the subsequent upload phase, our scheme needs a little more cost than scheme [8], [10], which is $8|\mathbb{G}| + (s + 5)|\mathbb{Z}_q|$. Because the subsequent uploader should regenerate the newly negotiate IBBE key and its public auditing key. In the data auditing phase, the communication cost is the same among schemes [8], [10] and ours, which is $4|\mathbb{Z}_q|$. While in scheme [9], they upload the challenge set content. Therefore, the communication is a little bigger, which is $(2c + 1)|\mathbb{Z}_q| + 3|\mathbb{G}|$. For batch auditing, the communication cost is related to the number of files $\mathcal{F}$ in the scheme [9]. In scheme [10], the communication cost is related to the number of challenge block numbers, while it is related to the number of sectors in our scheme.

Storage cost: Let $\mathcal{F}$ denote the number of files. Let $\mathcal{U}$ denote the number of users who owns the same file. The comparison result is shown in Table IV. In schemes [8] and [9], the storage cost is related to the number of users $\mathcal{U}$ and the number of blocks n. Specifically, the storage cost on the CSP side in the scheme [8] and [9] is $\mathcal{U}n|\mathbb{G}| + |F|$ and $\mathcal{U}(n + 1)|\mathbb{G}| + |F|$ respectively. Because each user has its own authenticators, and all authenticators should be stored. In our scheme, the private key of file is based the file instead the private key of the users. Thus, the authenticator is irrelevant to the users. The storage cost in our scheme is $(\mathcal{U} + w + 4)|\mathbb{G}| + (\mathcal{U} + 2)|\mathbb{Z}_q| + |F|$, which is greatly reduced. On the user side, it only needs to store some secret information. In the scheme [8], the user needs to store the secret key sk, so the storage cost is $|\mathbb{Z}_q|$. In scheme [9], the user needs to store the secret keys (x, ssk), and $\mathcal{F}n$ convergent keys, so the storage cost is $(2 + \mathcal{F}n)|\mathbb{Z}_q|$. In the scheme [10], the user needs to store the information t, t^*, sk for each file, so the storage cost is $(\mathcal{F} + 1)|\mathbb{Z}_q| + \mathcal{F}|\mathbb{G}|$. In our scheme, the user only needs to store the secret keys $(g, \gamma, sk_{ID}, ssk)$, which is $4|\mathbb{Z}_q| + |\mathbb{G}|$. It is a fixed value. Besides, some public information and data deduplication information is stored on the blockchain. In scheme [8], it needs to store $(c, k_1, k_2, \mu, \sigma)$ on-chain, so the storage cost is $4|\mathbb{Z}_q| + |\mathbb{G}|$. In scheme [9], it needs to store t, μ, σ, R on-chain, so the storage cost is $3|\mathbb{G}| + |\mathbb{Z}_q|$.

In scheme [10], it needs to store $(c, k_1, k_2, \mu, t, t^*, U, SSP)$ on-chain, so the storage cost is $5|\mathbb{Z}_q| + |\mathbb{G}|$. In our scheme, it needs to store $(c, k_1, k_2, \{\mu_j\}_{j=1}^{s}, \sigma)$ on-chain, so the storage cost is $(3+s)|\mathbb{Z}_q| + |\mathbb{G}|$. Our scheme's storage on-chain is bigger than other schemes, but the storage on the CSP side is reduced. The number of signatures is reduced as well.

B. Performance Analysis

The scheme was implemented based PBC Library.[4] All the experiments were executed on a host machine with Windows 10, Intel i7 2.5 GHz CPU, and 8 GB memory. The size of $\mathbb{G}$ and $\mathbb{Z}_q$ were set to 160 b, the size of a sector in one block was set to 160 b. All experiments were performed 1000 times and then the average results were obtained. Suppose χ blocks are corrupted in a file, the probability $P_c = \chi/n$ denotes the ratio of corruption. Let P_d denote the probability of detecting corrupted blocks, suppose there are $\overline{X}$ corrupted data blocks, the following equations holds

$$P_d = P\{\overline{X} \geq 1\} = 1 - P\{\overline{X} = 0\}$$
$$= 1 - \frac{n-\chi}{n} \times \frac{n-1-\chi}{n-1} \times \cdots \times \frac{n-c+1-\chi}{n-c+1}.$$

Since $\frac{n-\chi}{n} > \frac{n-c+1-\chi}{n-c+1}$, we have

$$1 - \left(\frac{n-\chi}{n}\right)^c < P_d < 1 - \left(\frac{n-c+1-\chi}{n-c+1}\right)^c.$$

That is, when $P_d = 99\%$ and $P_c = 1\%$, the required number of challenge blocks is $c = 460$; when $P_d = 95\%$ and $P_c = 1\%$, then $c = 300$. Thus, we set $c = 300$ and $c = 460$ respectively in the experiment.

Figs. 6 and 7 showed the computation cost of initial upload. The file size was set from $1\,MB$ to $10\,MB$ and the sector size from 10 to 100. As observed, our scheme had large advantages in generating authenticator tags compared with other three schemes. Especially, the computational overhead dropped sharply as the number of segments increases. Although the other three schemes were irrelevant to the number of sectors, the computational overhead of these schemes were generally higher than ours.

Fig. 8 showed the computation cost of subsequent upload and data auditing. Fig. 8(a) showed the computation overhead in PoW phase. The challenge block numbers were set to $c = 300$ and $c = 460$ respectively. From Fig. 8(a), the computation cost of proof generation in our scheme can be negligible. The computation cost of proof verification was a little small than the other two schemes. That is because we utilize the characteristic of add accumulation. Specifically, the subsequent uploader should generate $PoW_B.Proof = \{\{\mu_j\}_{1 \leqslant j \leqslant s}\}$, $\mu_j = \sum_{i=1}^{c} v_i c_{b_i j}$, the mainly cost involves addition and multiplication. Fig. 8(b) showed the computation overhead in the auditing phase. From Fig. 8(b), as observed, the computation cost had little different from other schemes.

Fig. 9 showed the computation overhead in the batch auditing phase. The number of batch files were set from 5 to 21 with an increment of 2 in each test. The challenge block numbers were set to 300 and 460 respectively. As observed, the computation cost increased with the number of auditing files. Our scheme had little difference with scheme [10].

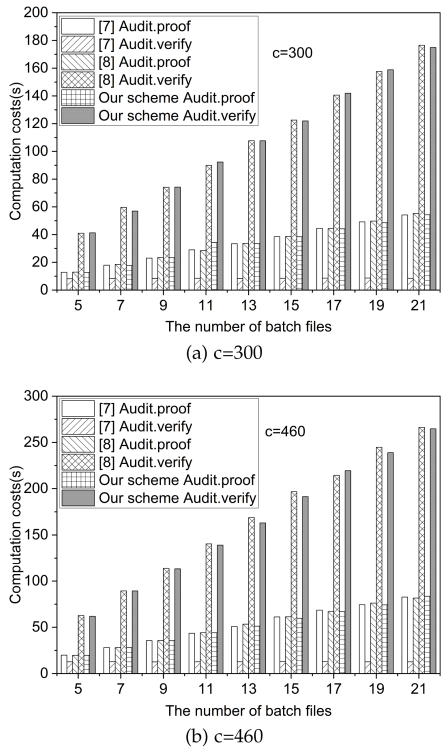


Fig. 9. Computation costs of batch auditing when c = 300 and 460.

Fig. 10 showed the storage costs. The file size was set to $1\,MB$. Fig. 10(a) and B showed the storage costs on the CSP side. In Fig. 10(a), the number of users who own the same file were set to $U = 50$. In additon, we define $w = \frac{n}{s} = 10$. The number of data blocks were set from 100 to 1000 with an increment of 100 in each test. As observed, the storage costs grew linearly with the number of growing data blocs in the scheme [8], [9], [10]. While in our scheme, it was a fixed value and smaller than that in other schemes. In Fig. 10(b), the number of users who own the same file was set from 0 to 100 with an increment of 10. The value of n and w were set to $n = 500$ and $w = 10$. As observed, the storage costs in three schemes except [10] grew linearly with the number of growing users. While in our scheme, it was a small value and grows slowly. In addition, it was smaller than the scheme [10]. Fig. 10(c) showed the storage costs on the user side. In Fig. 10(c), the number of files was set from 1 to 10 with an increment of 1. As observed, the storage costs in our scheme and scheme [8] were irrelevant to the number of files. That is because the user should store the private key for each file in scheme [9], [10]. While in our scheme, the user only stored the master secret key and the IBBE key. Fig. 10(d) showed the storage costs of the blockchain. In Fig. 10(d), we set the number of sectors from 1 to 10 with an increment of 1. As observed, only our scheme was relevant to the number of sectors. But the computation costs were greatly reduced.

[4]https://crypto.stanford.edu/pbc/download.html

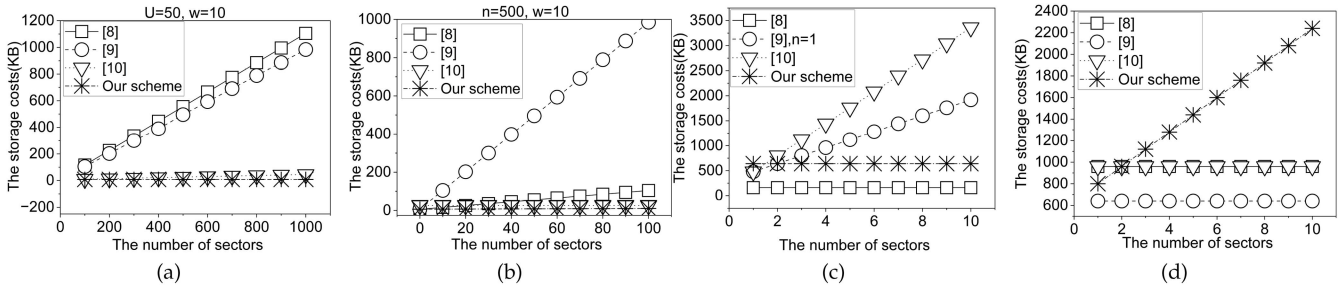


Fig. 10. Storage costs based (a) CSP side when $U = 50$ and $w = 10$ with the number of data blocks; (b) CSP side when $n = 500$ and $w = 10$ with the number of users own the same file; (c) User side with the number of files; (d) Blockchain side with the number of sectors.

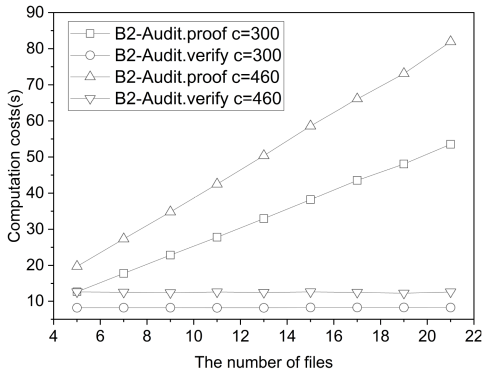


Fig. 11. Many users with the same file auditing.

Fig. 11 showed the many users with the same file auditing. Only our scheme supported this function. Similarly, the challenge block numbers were set to $c = 300$ and $c = 460$ respectively. The number of files was set from 5 to 21 with an increment of 2. As observed, the proof generation computation costs grew linearly with the number of growing auditing files. While the proof verification computation costs grew slowly.

VIII. Conclusion

Focusing on deduplication, including key deduplication and authenticator deduplication, we proposed a blockchain-based shared data integrity auditing and deduplication scheme. Specifically, we utilized ID-based broadcast encryption to achieve key deduplication on the user side. We utilized the characteristic of convergent encryption to achieve authenticator deduplication on the CSP side. In addition, we achieved decentralized data integrity auditing based blockchain, which can avoid a single point of failure and improve the credibility of the auditing result. On this basis, we designed two bath auditing protocols for different scenarios. Theory and experiment showed that the proposed scheme was significant in reducing computation overhead and saving storage space.

References

[1] Y. Ji, B. Shao, J. Chang, M. Xu, and R. Xue, "Identity-based remote data checking with a designated verifier," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–14, 2022.

[2] X. Zhang, X. Wang, D. Gu, J. Xue, and W. Tang, "Conditional anonymous certificateless public auditing scheme supporting data dynamics for cloud storage systems," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 5333–5347, Dec. 2022.

[3] 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. 2023.

[4] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. Decentralized Business Review, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[5] J. Bernabe, J. Canovas, J. Hernandez-Ramos, R. Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: Review and challenges," *IEEE Access*, vol. 7, pp. 164908–164940, 2019.

[6] K. Gai, J. Guo, L. Zhu, and Shui S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2009–2030, Third Quarter 2020.

[7] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3548–3558, Jun. 2019.

[8] 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. 2021.

[9] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and S. Willy, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, 2020.

[10] 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.

[11] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2013, pp. 296–312.

[12] J. Douceur, A. Adya, W. 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.

[13] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Jun. 2014.

[14] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Toward encrypted cloud media center with secure deduplication," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 251–265, Feb. 2017.

[15] S. Li, C. Xu, and Y. Zhang, "CSED: Client-side encrypted deduplication scheme based on proofs of ownership for cloud storage," *J. Inf. Secur. Appl.*, vol. 46, pp. 250–258, 2019.

[16] P. Singh, N. Agarwal, and B. Raman, "Secure data deduplication using secret sharing schemes over cloud," *Future Gener. Comput. Syst.*, vol. 88, pp. 156–167, 2018.

[17] X. Gao et al., "Achieving low-entropy secure cloud data auditing with file and authenticator deduplication," *Inf. Sci.*, vol. 546, pp. 177–191, 2021.

[18] J. Bai, J. Yu, and X. Gao, "Secure auditing and deduplication for encrypted cloud data supporting ownership modification," *Soft Comput.*, vol. 24, no. 16, pp. 12197–12214, 2020.

[19] G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

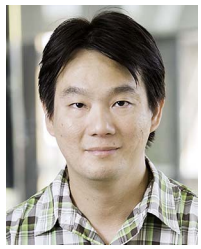
[20] Y. Li and F. Zhang, "An efficient certificate-based data integrity auditing protocol for cloud-assisted WBANs," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11513–11523, Jul. 2022.

[21] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 356–365, Jan.–Mar. 2022.
[22] J. Gudeme, S. Pasupuleti, and R. Kandukuri, "Certificateless privacy preserving public auditing for dynamic shared data with group user revocation in cloud storage," *J. Parallel Distrib. Comput.*, vol. 156, pp. 163–175, 2021.
[23] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 14–24, Feb. 2022.
[24] Y. Miao, Q. Huang, M. Xiao, and H. Li, "Decentralized and privacy-preserving public auditing for cloud storage based on blockchain," *IEEE Access*, vol. 8, pp. 139813–139826, 2020.
[25] X. Yang, X. Pei, M. Wang, T. Li, and C. Wang, "Multi-replica and multi-cloud data public audit scheme based on blockchain," *IEEE Access*, vol. 8, pp. 144809–144822, 2020.
[26] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, pp. 1–16, 2019.
[27] J. Shu, X. Zou, X. Jia, W. Zhang, and R. Xie, "Blockchain-based decentralized public auditing for cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2366–2380, Oct.–Dec. 2022.
[28] 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.
[29] Y. Zhang, C. Xu, N. Cheng, and X. Shen, "Secure password-protected encryption key for deduplicated cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2789–2806, Jul./Aug. 2022.
[30] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted Big Data in cloud," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 138–150, Jun. 2016.
[31] X. Yu, H. Bai, Z. Yan, and R. Zhang, "VeriDedup: A verifiable cloud data deduplication scheme with integrity and duplication proof," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 680–694, Jan./Feb. 2022.
[32] L. Liu, Y. Zhang, and X. Li, "KeyD: Secure key-deduplication with identity-based broadcast encryption," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 670–681, Apr.–Jun. 2021.
[33] Z. Yang, J. Li, Y. Ren, and P. Lee, "Tunable encrypted deduplication with attack-resilient key management," *ACM Trans. Storage*, vol. 18, no. 4, pp. 1–38, 2022.
[34] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2013, pp. 145–153.
[35] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016.
[36] 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 INFOCOM Conf. Comput. Commun.*, 2017, pp. 1–9.
[37] Q. Zhang, D. Sui, J. Cui, C. Gul, and H. Zhong, "Efficient integrity auditing mechanism with secure deduplication for blockchain storage," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2365–2376, Aug. 2023.
[38] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2007, pp. 200–215.
[39] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, 2011, pp. 491–500.
[40] Y. Deswarte, J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Proc. Work. Conf. Integrity Intern. Control Inf. Syst.*, 2003, pp. 1–11.
[41] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2008, pp. 90–107.

Ying Miao received the MS degree from South China Agricultural University. She is currently working toward the PhD degree with the School of Computer Science and Technology, Beijing Institute of Technology. She has authored or coauthored more than ten papers about blockchain, data security and machine learning. Her research interests include information security, blockchain, and cloud computing.

Keke Gai (Senior Member, IEEE) received the PhD degree in computer science from Pace University, New York, NY, USA. He is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. He has authored or coauthored more than 170 peer-reviewed papers in recent years. His research interests include cyber security, blockchain, AI security, and privacy computation. He was the recipient of ten Best Paper Awards. He is also the editor-in-chief of journal Blockchains and the Editorial Board member of a few journals, including *IEEE Transactions on Dependable and Secure Computing, Journal of Parallel and Distributed Computing*, and *Future Generation Computer Systems*. He is also with few academic organizations, such as a co-chair of IEEE Technology and Engineering Management Society (TEMS)'s Technical Committee on *Blockchain* and *Distributed Ledger Technologies*. He is a Senior Member of IEEE.

Liehuang Zhu (Senior Member, IEEE) received the BE and ME degrees from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the PhD degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2004, He is currently a professor with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He has authored or coauthored more than 100 peer-reviewed journal or conference papers, including more than ten IEEE/ACM Transactions papers (*IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Smart Grid*, *Information Sciences*, *IEEE Network*, and *Computer and Security*). His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing. He was the recipient of the number of IEEE Best Paper Awards, including IWQoS 17' and TrustCom 18'.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the PhD degree in information security from the Queensland University of Technology, Australia, in 2006. He is currently holding the Cloud Technology Endowed Professorship with The University of Texas at San Antonio. He is also the founding co-editor-in-chief of *ACM Distributed Ledger Technologies: Research & Practice*, and the founding chair of IEEE Technology and Engineering Management Society Technical Committee (TC) on *Blockchain* and *Distributed Ledger Technologies*. He was the recipient of the 2022 IEEE Hyper-Intelligence TC Award for Excellence in Hyper-Intelligence Systems (Technical Achievement Award), 2022 IEEE TC on Homeland Security Research and Innovation Award, 2022 IEEE TC on Secure and Dependable Measurement Mid-Career Award, and the 2019 IEEE TC on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher).

Jaideep Vaidya (Fellow, IEEE) received the BE degree in computer engineering from the University of Mumbai, India, the MS and PhD degrees in computer science from Purdue University, USA. He is currently a distinguished professor of computer information systems with Rutgers University, USA, director of the Rutgers Institute for Data Science, Learning, and Applications, and acting chair with the Department of Management Science & Information Systems. He has authored or coauthored more than 200 technical papers in peer-reviewed journals and conference proceedings. His research interests include security, privacy, data mining, and data management. He was the recipient several Best Paper awards from the premier conferences in data mining, databases, digital government, security, and informatics and notably, the team he led won the first prize in the US-UK Privacy Enhancing Technologies Challenge in the Financial Crime Track. He is also an ACM distinguished scientist, IEEE and AAAS Fellow and was the editor in chief of the *IEEE Transactions on Dependable and Secure Computing*.