# Blockchain-assisted Searchable Integrity Auditing for Large-Scale Similarity Data with Arbitration

Ying Miao, Keke Gai, *Senior Member, IEEE,* Yu-an Tan, Liehuang Zhu, *Senior Member, IEEE,* and Weizhi Meng, *Senior Member, IEEE*

***Abstract*—Data integrity auditing technology serves as an essential tool to ensure the data's integrity with the popularity of remote storage. However, existing data integrity auditing models are unsuitable for a large number of files with inter-relationships and heavily depend on a centralized Third-Party Auditor (TPA). To address these issues, in this paper we propose a blockchain-assisted searchable integrity auditing scheme for large-scale similarity data. To broaden the scope of the auditing model and enhance its ability to handle interconnected files, we utilize the keyword to design a search index and a trapdoor to achieve authenticator searchability for the interconnected files. The integrity of the searching result from the cloud side can be guaranteed at the same time. To reduce reliance on centralized TPA and enhance the credibility and transparency of auditing, we integrate blockchain technology along with smart contracts to replace TPA and achieve multitask auditing. We adopt a certificateless cryptosystem to generate the authenticator, while considering the cost reduction. Moreover, an arbitrator is proposed to achieve fairness judge. Theoretical and security analysis demonstrate that the proposed scheme is efficient and secure, making it a promising solution for data auditing in a wide range of applications.**

***Index Terms*—Blockchain, Integrity auditing, Searchability, Similarity data, Arbitration.**

## I. Introduction

THE emergence of cloud computing has dramatically transformed the paradigm of data-driven applications to centralized-based services via offering a series of technical merits, such as hidden complexity, flexibility, scalability, and cost-effectiveness [1]. Enterprises have a chance to launch a complete IT (information technology) solution without concerning about technical obstacles, from hardware to software. However, one of the long-standing issues in cloud computing implementations is that *Data Owners* (DOs) lack controls on their outsourcing data, which raises security and privacy concerns. Among multiple technical dimensions in addressing control lacks, data integrity and availability are considered two fundamental security requirements, such that *Provable Data Possession* (PDP) has emerged in this context [2]–[4].

PDP allows DO to verify the integrity of remote data without the need to download it from the *Cloud Server* (CS). For convenience, DO does not need to be online all the time; instead, a *Third Party Auditor* (TPA) is delegated to facilitate validation on behalf of DO. A typical three-party (DO, CS, TPA) PDP scheme consists of a few key components [3]. In general, authenticators are generated by different cryptosystems, e.g. *Public Key Infrastructure*-based (PKI) cryptosystem [5]–[7], identity-based cryptosystem [8]–[11], certificateless-based cryptosystem [11]–[14]. PKI can lead to increased certificate management costs, while identity-based cryptosystems can raise secret key management expenses. Certificateless-based cryptosystems offer a solution to these challenges. In challenge phase, two major methods of challenge information generation from TPA are generating challenge seeds [15] and sets [5], [11], [16]. In proof generation and verification phases, privacy protection is a concern of existing schemes [8], [17], [18]. However, there are some limitations to the existing schemes according to our investigations.

The main limitation is that current PDP technology cannot achieve searchability, especially when dealing with large-scale similarity data. An estimation from the Internet Data Center indicates that the personal data will reach 5200 GB in 2020 [1]. Large-scale similarity data sets can involve millions of high-dimensional vectors or data with similarity relationships, making it computationally intensive and impractical for existing PDP approaches to efficiently verify their possession. Traditional PDP schemes are designed for exact data integrity checking, meaning they verify that specific, unaltered data blocks exist on the server. These schemes don't naturally support operations that involve approximate matching or semantic similarity, which are core requirements when working with similarity data. For individual, the DO may only be concerned with the integrity of certain specific files, such as audio data or video data. For research institutions, researchers focus on data in a particular field, such as heartbeat data or blood routine data. Therefore, there is a strong and urgent need to implement searchable auditing. Although some schemes, which utilize keywords to achieve searchability, have been proposed [17], [19]. These schemes rely on PKI, which imposes a substantial burden in terms of certificate management costs and exists some other challenges.

Another limitation is the reliance of the third party as the system assumes the trustworthiness of TPA, making it the key of audit success or failure [3]. Any unexpected activities by TPA can result in undesirable consequence. Even though some studies have made attempts to enhance the credibility

Y. Miao, K. Gai, Y. Tan and L. Zhu are with School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, 100081, China (e-mail: {yingmiao, gaikeke, tan2008, liehuangz}@bit.edu.cn).

W. Meng is with the School of Computing and Communications, Lancaster University, United Kingdom. Email: weizhi.meng@ieee.org.

This work is partially supported by the National Natural Science Foundation of China (Grant No. U24B20146, No. 62232002, No. 62372044).

Corresponding author: Keke Gai (gaikeke@bit.edu.cn)

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

and transparency of the auditing process, such as through the exploration of blockchain-based solutions, the reliance on TPA persists [20]–[23]. In order to audit data, TPA must remain online and continuously await responses from CS. This requirement not only amplifies the complexity and costs associated with auditing but also causes a risk of audit ineffectiveness when TPA fails in responding to auditing requests promptly. Hence, there is an urgent need to develop a solution that can ensure the reliability of auditing results independent of TPA.

To enhance the applicability of the auditing model, the important thing is to clarify the relationships among these files first. Naturally, an index can be established to achieve searchability. In this approach, a DO provides a search trapdoor to CS, allowing CS to find the corresponding files based on the trapdoor and then conduct the auditing. However, there are a few challenges when designing a searchable PDP.

- **Honest Execution of Searchable Auditing Issue**. It is a challenge to guarantee that CS performs searchable auditing honestly while ensuring that all files meeting the requirements participate in the auditing. Instead of transmitting all data requiring auditions to CS, a searchable function can be executed at the CS side through a search trapdoor. However, a search trapdoor used for auditing only contains indeterminate information about the data (e.g., file numbers), such that the CS probably use partial files rather than the entire file for auditing.
- **Auditing Credibility without TPA Issue**. Ensuring the credibility of the auditing results without TPA is a challenge. To minimize reliance on TPA, the verification task is performed by the CS. As the entity being challenged, it is a challenge to ensure that the CS conducts the auditing task honestly and provides the trustworthy results.
- **Fairness of Dispute Resolution Issue**. A challenge exists in guaranteeing the reliability of the judge results when disputes happen between DO and CS. Despite blockchain increasing the transparency of auditing results, disputes may exist when data loss happen but CS does not generate correct report on the blockchain. Another situation is that the data might be stored correctly but a DO charges for economic losses for data corruption.

In this paper, instead of relying on TPA, we propose a searchable data integrity auditing scheme by leveraging the category of files. Our goal is to expand the scope of applications for auditing models. We employ certificateless cryptography as a tool to construct the authenticator. The main contribution of this paper can be outlined as follows:

1) To enhance the application wide of auditing model, we propose a searchable data integrity auditing scheme for large-scale similarity data. To achieve searchability, we utilize the keyword to design a search index and a trapdoor to achieve authenticator searchability and matching. When dealing with a large number of interrelated files auditing, DO just provides a search trapdoor, CS conducts searches, matches and offers the relevant proof information for auditing. The integrity of the search result can be guaranteed at the same time.
2) To enable searchable auditing and ensure the security of the auditing process, we design the index matrix and index structure. The index structure serves a dual purpose by safeguarding file privacy and facilitating auditing, thus ensuring the security of the overall auditing process. To reduce the certificate and key management costs, we utilize certificateless cryptography in the authenticator generation phase.
3) To enhance the credibility and transparency of the auditing, we leverage blockchain and smart contracts instead of TAP to participate in the auditing process. Specifically, challenge information is generated in a decentralized manner, making the auditing process public, tamper-proof, and traceable. Additionally, multiple auditing tasks are initiated through smart contracts in one time. When the auditing task is published by DO, the task is executed and verified as mandated by CS. This approach ensures that the auditing process is traceable, and any improper behavior can be easily detected. An arbitration process is proposed to deal with conflicts between DO and CS.
4) We provide the detailed security analysis that demonstrates the proposed scheme's compliance with auditing soundness. Additionally, We also conduct the experimental simulation. The experimental results show that the proposed scheme is efficient in authenticator generation phase and the index generation phase.

The paper is structured as follows: We explore previous research in PDP and PoR in Section II. We provide essential foundational knowledge and problem formulation to establish a clear understanding of the proposed scheme in Section III and Section IV. Section V elaborates on the details of the proposed scheme. We conduct a security analysis in Section VI and performance analysis in Section VII. We conduct a disscussion and draw a conclusion in Section VIII and IX.

## II. Related Work

To guarantee the integrity of the outsourced data and without downloading the whole data, Ateniese *et al.* [2] and Juels *et al.* [26] introduced the concepts of PDP and *Proof of Retrievability* (PoR). Subsequently, various data integrity verification techniques, including those related to data privacy [7], data updating [27], data access control [9] and distributed data auditing [11], have been developed. To address various requirements and functionalities, numerous data integrity auditing schemes have been proposed using different cryptography techniques. TABLE I listed partial recent related work.

To achieve data sharing among group and protect the group user's privacy, Hu *et al.* [16] adopted group signatures to establish an auditing scheme with anonymity function. In the scheme, the group manager possessed knowledge of and control over user identities. In order to minimize certificate management expenses, Li *et al.* [28] employed identity-based signatures to develop an auditing scheme tailored for a multi-copy and multi-cloud environment. To reduce the tree's storage costs, Guo *et al.* [5] utilized a single authenticated tree shared among multiple replicas and achieved batch verification for multiple challenges. To protect different types of data and

TABLE I: Comparison of Provable Data Possession Schemes

| Schemes | Years | Auth-type | Participants | Merits | Limitations |
|---|---|---|---|---|---|
| [24] | 2015 | Certificate-less | DO, CS, TPA, BC | Reduce the burden of certificate management and key management, and without the strong assumption that TPA is honest and reliable. | Cannot resist challenge information guessing attack; The TPA should always keep online. |
| [8] | 2019 | Identity | DO, CS, TPA, BC | Reduce the burden of certificate management and without the strong assumption that TPA is honest and reliable. | Bear the burden of key management and cannot resist challenge information guessing attack; The TPA should always keep online. |
| [5] | 2020 | PKI | DO, CS, TPA | Achieve single authenticated tree across multiple replicas and batch verification. | Cannot resist repeat challenge attack; The TPA should always keep online. |
| [17] | 2021 | PKI | DO, CS, TPA | Achieve searchable data integrity checking. | Cannot resist the same keyword summation attack; The TPA should always keep online. |
| [6] | 2021 | PKI | DO, CS | Locate the CS and resist denial of service attack. | The DO should always keep online; The TPA should always keep online. |
| [12] | 2022 | Certificate-less | IoT device, CS, fog node, BC | Achieve reliable checking and auditing fairness. | The challenge information is controlled by CS. |
| [10] | 2022 | Identity | DO, CS, TPA | The DO need not to store original data blocks on the cloud, save the storage overhead. | Rely on the strong assumption that TPA is honest and reliable; The TPA should always keep online. |
| [13] | 2023 | Chaotic system | DO, TPA, BC | The auditing task cannot depend on challenging the CS, reduce the computation time by a new audit tree and achieve fair payments. | Cannot support batch verification; The TPA should always keep online. |
| [25] | 2023 | PKI | DO, BC, key server | Achieve transparent auditing and authenticator deduplication, resist exfiltration. | The challenge information is controlled by CS indirectly, cannot resist challenge information guessing attack. |

facilitate data sharing, Liu *et al*. [7] utilized the property of sanitizable signature to construct an auditing scheme. Yang *et al*. [10] designed an identity-based signature and utilized algebraic operations for auditing in resource-limited cloud storage environments, reducing the storage, communication and computation costs. To achieve data dynamics and protect identity privacy, Zhang *et al*. [14] utilized certificate-less signature in combination with blind technology to achieve conditional identity privacy preservation. Malicious user behavior can be inferred. To achieve anonymity of user identities, DO in [18] generated block authenticators using ring signature technology. However, this approach increased the verification costs, as TPA needed to have access to all identities in the ring. In efforts to enhance the efficiency of batch auditing, Shen *et al*. [29] leveraged the properties of algebraic signatures to accelerate the efficiency of signature aggregation.

However, many schemes rely on a centralized TPA with a strong assumption that TPA is honest. Since TPA may not always be honest and could collude with CS, generating biased challenge information. To safeguard against data analysis in centralized cloud data and pinpoint data faults, Su *et al*. [6] introduced a decentralized self-auditing scheme designed for a multi-cloud environment that did not depend on a TPA. Additionally, some integrity auditing work based on blockchain has been proposed. Zhang *et al*. [24] introduced a decentralized data integrity checking scheme that leveraged a sequence of public block hashes of block headers as seeds to construct challenge information, eliminating the need for a TPA for this task. Xue *et al*. [8] leveraged identity-based cryptosystem combined with blockchain technology to design a decentralized auditing scheme. In the scheme, they made use of a set of nonce values from the blockchain to form the challenge seeds. However, the scheme [8], [24] cannot resist challenge information guessing attack due to the transparency and publicity of blockchain, where CS can potentially obtain the challenge information in advance. Zhao *et al*. [22] utilized lifted EC-ElGamal cryptosystem and blockchain to achieve public batch auditing. The auditing information was stored on the blockchain, ensuring the integrity and unforgeability of tag information. However, CS was required to always provide the whole ciphertext for the auditor, leading to increased communication costs. Yuan *et al*. [30] utilized smart contract to achieve fair arbitration and penalize untrustworthy entities. Fan *et al*. [31] proposed a data integrity framework for edge computing based on blockchain, designed to meet the demands of dynamic networks. Li *et al*. [20] utilized blockchain to construct a transparent deduplication and integrity auditing scheme, which eliminated the need for a TPA and reduced communication costs for the DO. Zhou *et al*. [12] employed blockchain to record the auditing results and resisted dishonest fog nodes. They also designed a dynamic structure to reduce the computation costs of tags from IoT devices. Focus on the security issue in existing smart contract-based public auditing, Li *et al*. [25] introduced a whistleblower, which timely notified DO of any corruption, to monitor the information on the blockchain. However, the challenge information in [12], [25] was controlled by CS, which could not guarantee the randomness of the challenge information. Wang *et al*. [13] designed reward and punishment rules based on smart contracts. Additionally, they utilized chaotic systems to achieve non-interactive integrity auditing. To reduce the costs of certificate and key management, Miao *et al*. [23] utilized certificate-less signature combined blockchain to achieve multiple copy and multiple cloud auditing, which utilized smart contracts to facilitate data files, data blocks and CS faults localization. In schemes relying on TPA, it is imperative for TPA to maintain continuous online availability. Most recently, Gao *et al.* [17] considered the relationships among files and achieved keyword searchable auditing. To hide the audit frequency, Xue *et al.* [19] utilized bloom filters to achieve fuzzy matching and audited files based on keywords. However, the index structure in [17], [19] cannot resist the same keyword files summation attack.

Blockchain technology integrates various components such as chain-based data structures, *peer-to-peer* (P2P) networking, consensus mechanisms, cryptographic techniques, and smart contracts to establish a decentralized and distributed ledger system [32]. Many key challenges inherent in centralized systems—such as trust deficits, lack of transparency, and vulnerability to tampering can be resolved through blockchain technology [33], [34]. With the advancement of blockchain technology, smart contracts play a pivotal role in constructing decentralized applications and ensuring transparent functional modules [35]. The development of smart contracts has catalyzed extensive research, notably in security [36], scalability [37], and decentralized governance [38].

## III. Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_T$ be additive cyclic groups with the prime order $q$. Let $a$ and $b$ be random elements in $\mathbb{Z}_q^*$. Let $g$ be a generator of group $\mathbb{G}$.

### A. Bilinear Map

The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ meets the following three properties:

- Bilinearity: $\forall u, v \in \mathbb{G}, a, b \in \mathbb{Z}_q^*$, $e(u^a, v^b) = e(u, v)^{ab}$ holds.
- Computability: $\forall u, v \in \mathbb{G}, a, b \in \mathbb{Z}_q^*$, it is efficient to compute $e(u^a, v^b)$.
- Non-degeneracy: $\exists u, v \in \mathbb{G}$, such that $e(u, v) \neq 1_T$.

### B. Hard Problems

**Definition 1.** (Discrete Logarithm (DL) Assumption). Given $(g, g^a) \in \mathbb{G}$, it is computationally intractable to retrieve the value $a$. For any *Probabilistic Polynomial Time* (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ to solve the DL problem is negligible. That is,

$$Adv_{\mathcal{A}}^{DL} = Pr[\mathcal{A}(g, g^a, g^b) \to a \in \mathbb{Z}_q^*] \leqslant \varepsilon.$$

**Definition 2.** (Computational Diffie-Hellman (CDH) Assumption). Given $(g, g^a, g^b) \in \mathbb{G}$, it is computationally intractable to retrieve the value $g^{ab}$, where $a$ and $b$ are unknown values. For any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ to solve the CDH problem is negligible. That is,

$$Adv_{\mathcal{A}}^{CDH} = Pr[\mathcal{A}(g, g^a, g^b) \to g^{ab} \in \mathbb{G}] \leqslant \varepsilon.$$

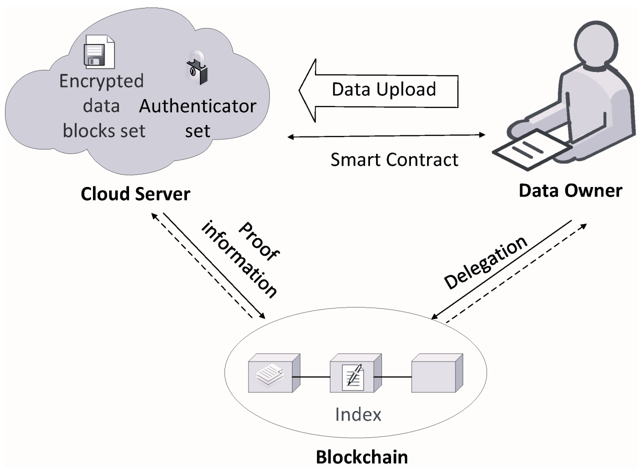


Fig. 1: System Model.

## IV. Problem Formulation

### A. System Model

The system model (shown in Fig. 1) consist of three entities: *Cloud Server* (CS), *Data Owner* (DO) and Blockchain (BC).

1) **Cloud Server (CS)**: CS has unlimited storage resources and computing power, and provides the storage and data integrity auditing for DO.
2) **Data Owner (DO)**: DO has some files to be outsourced. DO splits each file into blocks, extracts some keywords for each file, generates a secure index for these files and uploads these information into the cloud.
3) **Blockchain (BC)**: BC provides a public and immutable platform for the data integrity auditing process. The smart contract records the basic information of the data, generates decentralized and unpredictable challenge information as challenge and records the proof information as evidence for checking.

Combined with Fig. 1, the system is outlined by the followings. Suppose there are $v$ files to be stored. DO splits each file into $n$ blocks, encrypts these blocks and generates the authenticator for these blocks. Then DO extracts some keywords for each file and builds a secure index for these files. Finally, DO uploads the encrypted data block set, the index set and the authenticator set into the cloud for storage. Next, DO and CS agree on data integrity checks. Subsequently, DO delegates the task and establishes a smart contract on the blockchain. CS proceeds to conduct the data integrity audit based on the challenge details in smart contract and subsequently uploads the proof information onto the blockchain, serving as substantiating evidence.

### B. Threat Model

CS might attempt to access the content of the data since it has outsourced using the keyword and the search trapdoor. It could also try to create a fraudulent authenticator for the outsourced data to deceive the verification process. Additionally, CS may attempt to replace the proof information with that of different challenge data to pass the verification.

## C. Design Goals

The proposed scheme meets the following design goals:

- **Searchable-auditing soundness**: DO can perform the integrity checking of all similar data, CS can perform the auditing information and verification correctly.
- **Transparent-auditing**: the challenge information and the auditing process information can be traced from blockchain.
- **Multitask**: multiple audit task content are generated a time in a decentralized way by DO with the help of smart contract. CS performs the auditing task one by one according to the plan.
- **Arbitration**: When dispute happen between DO and CS, an arbitrator deals with the dispute and finds the dishonest party.

## D. Definition

**Definition 3.** *A blockchain-assisted multi-task and searchable data integrity auditing scheme consists of nine algorithms:*

1) $(params, msk, P_0) \leftarrow SysIni(\kappa)$: When provided with a security parameter $\kappa$, the algorithm generates the system outputs $params$ and the master secret/public key pair $(msk, P_0)$ as its output.
2) $(\{A_k = g^{a_k}\}_{k=1}^{s}, V) \leftarrow FileIni(\{F_v\})$: When supplied with the file set $\{F_i\}_{i=1}^{v}$ as input, the algorithm outputs the public values $\{A_k = g^{a_k}\}_{k=1}^{s}$ and the initial index vector $V$ as its output.
3) $(sk_{ID}, x, P_1) \leftarrow KeyGen(ID, msk)$: When provided with the user's identity $ID$ and master secret key $msk$, the algorithm yields the private key $(sk_{ID}, x)$ and the public key $P_1$ associated with that user.
4) $I \leftarrow IndexGen(x, W, V)$: When provided with the secret key, the keyword set, and the index vector $(x, W, V)$ as input, the algorithm produces a index vector $I$.
5) $\Phi \leftarrow AuthGen(\{F_i\}_{i=1}^{v}, x, sk_{ID})$: When provided with the file set $\{A_k = g^{a_k}\}_{k=1}^{s}$ and the private key $(sk_{ID}, x)$ as input, the algorithm generates the authenticator set $\Phi$.
6) $(T_{w'}, n_{w'}) \leftarrow TrapdoorGen(w')$: Upon receiving the keyword $w'$, the algorithm produces the search trapdoor and the value $(T_{w'}, n_{w'})$.
7) $Chal \leftarrow ChalGen(T_{w'})$: Given the search trapdoor $w'$ as input, the algorithm produces the updated search trapdoor $T'_w$ and the value $n_{w'}$.
8) $(Proof, S_{w'}) \leftarrow ProofGen(Chal, I, \Phi)$: When provided with the challenge, the secure index, and the authenticator set $(Chal, I, \Phi)$, the algorithm generates the auditing proof $Proof$ and the challenge index information set $S_{w'}$.
9) $(true/false) \leftarrow SelfVerify(Chal, Proof)$: When presented with the challenge information and the auditing proof $(Chal, Proof)$, the algorithm produces an auditing result, either $true$ or $false$.

## E. Security Model

There exist three adversaries $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. We design several games among the challenger $\mathcal{C}$ and adversaries $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$. $\mathcal{A}_1$ can replace DO's public key but doesn't have access to the master secret key. $\mathcal{A}_2$ can acquire the master secret key but cannot replace DO's public key. $\mathcal{A}_3$ attempts to create a fraudulent integrity proof to deceive the verification process.

$Game1$: This game involves $\mathcal{C}$ and $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$.

**SysIni, FileIni**: The challenger $\mathcal{C}$ invokes the **SysIni** and **FileIni** algorithm and retrieves public parameters $params$, the master key, public values and the initial index vector. $\forall \mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$, $\mathcal{C}$ sends the $params$, public values and the initial index vector to $\mathcal{A}$. If $\mathcal{A} = \mathcal{A}_2$, $\mathcal{C}$ also posts the master secret key to $\mathcal{A}$.

**Query Phase**: $\forall \mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ makes the following queries.

- $Hash\,Queries$: $\mathcal{A}$ makes hash queries for $\mathcal{C}$. $\mathcal{C}$ answers the value to $\mathcal{A}$.
- $Secret\,Key\,Query$: $\mathcal{A}$ queries the secret key of adaptively selected identity $ID_o$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}$.
- $Public\,Key\,Query$: $\mathcal{A}$ makes query for the the public key of adaptively selected identity $ID_o$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}$.
- $Public\,Key\,Replacement$: $\mathcal{A}_1$ has the power to replace the public key of identity $ID$ with any value.
- $Authenticator\,Query$: $\mathcal{A}$ makes query for the authenticator of the data block $(ID_i, j)$ for $F_i$. $\mathcal{C}$ executes the **AuthGen** algorithm to generate the corresponding authenticator to $\mathcal{A}$.

**Forgery Phase**: $\forall \mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ outputs a forgery authenticator $\sigma'_{ij}$ for $(ID'_i, j')$. If the following situation holds:

- The forged authenticator is valid.
- The identity $ID'$'s private key was not queried before, and its public key was not replaced.
- The authenticator of block $(ID'_i, j')$ was not be queried before.

**Definition 4.** If there is no *Polynomial Probability Time* (PPT) adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ who wins the $Game\,1$ with negligible probability, the authenticator is unforgeable.

$Game\,2$: This game involves $\mathcal{C}$ and $\mathcal{A}_3$.

**SysIni, FileIni**: The challenger $\mathcal{C}$ invokes the **SysIni** and **FileIni** algorithm and retrieves public parameters $params$, the master key, public values and the initial index vector. $\mathcal{C}$ sends the $params$, public values and the initial index vector to $\mathcal{A}_3$.

**Query Phase**: $\mathcal{A}_3$ makes the following queries.

- $Hash\,Queries$: $\mathcal{A}_3$ makes hash queries for $\mathcal{C}$. $\mathcal{C}$ answers the corresponding value to $\mathcal{A}_3$.
- $Public\,Key\,Query$: $\mathcal{A}_3$ queries the the public key of adaptively selected identity $ID_o$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_3$.
- $Authenticator\,Query$: $\mathcal{A}_3$ makes query for the authenticator of the data block $(ID_i, j)$ for $F_i$. $\mathcal{C}$ executes the **AuthGen** algorithm to generate the corresponding authenticator to $\mathcal{A}_3$.
- $TrapdoorGen\,Query$: $\mathcal{A}_3$ queries the trapdoor of keyword $w'$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_3$.
- $ChalGen\,Query$: $\mathcal{A}_3$ queries the challenge information. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_3$.

- $ProofGen\,Query$: $\mathcal{A}_3$ queries the proof information. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_3$.

**Forgery Phase**: $\mathcal{A}_3$ outputs a forgery proof information. If the following situation holds:

- The forgery proof information can pass the verification.
- The challenge information which was utilized to generate the proof information was not be queried.
- The forgery proof information was not be queried before.

**Definition 5.** If there is no Polynomial Probability Time (PPT) adversary $\mathcal{A}_3$ who wins the $Game\,2$ with negligible probability, CS performs the auditing correctly.

$Game\,3$: This game involves $\mathcal{C}$ and $\mathcal{A}_4$. $\mathcal{A}_4$ represents semi-honest CS and does not know secret key and pseudo-random function key and pseudo-random permutation key from DO. The indistinguishability of the trapdoor is to prevent attackers from getting keyword information out of the given trapdoor. The **SysIni** and **FileIni** phase are the same as defined in $Game\,2$.

**Phase 1**: $TrapdoorGen\,Query$: $\mathcal{A}_4$ queries the trapdoor of keyword $w'$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_4$.

**Challenge**: $\mathcal{A}_4$ chooses two keywords $\{w_0^*, w_1^*\}$ and sends it to $\mathcal{C}$. $\mathcal{C}$ randomly flips a coin, designating heads (heads-up) as 1 and tails (tails-down) as 0. Using the outcome of the coin toss, $\mathcal{C}$ executes the trapdoor generation algorithm to produce the trapdoor $T_{w_\beta} = \{\pi_o(w_\beta), f_l(\pi_o(w_\beta))\}$. $\mathcal{C}$ executes the index generation algorithm to produce the index $I_\beta = \{\pi_o(w_\beta), ev_{\pi_o(w_\beta)}, \Omega_{\pi_o}(w_\beta)\}_{k=1,2,\cdots,m}$. Finally, $\mathcal{C}$ sends them to $\mathcal{A}_4$.

**Phase 2**: $TrapdoorGen\,Query$: $\mathcal{A}_4$ queries the trapdoor of keyword $w'(w' \notin \{w_0^*, w_w^*\})$. $\mathcal{C}$ responds the corresponding value to $\mathcal{A}_4$.

**Guess**: $\mathcal{A}_4$ outputs $\beta' \in \{0,1\}$. If $\beta' = \beta$, $\mathcal{A}_4$ wins the $Game\,3$.

**Definition 6.** *The advantage of $\mathcal{A}_4$ to successfully distinguish the trapdoor is $Adv_{\mathcal{A}_4}^{\mathcal{C}} = |Pr[\beta' = \beta] - \frac{1}{2}|$. For any PPT adversary $\mathcal{A}_4$, if the advantage $Adv_{\mathcal{A}_4}^{\mathcal{C}}$ is negligible, the scheme satisfies trapdoor and index indistinguishabilitly.*

## V. The Construction

### A. Index Design

Suppose there are $v$ files to be stored. Each file are splited into $n$ blocks. DO begins extracting a set of $m$ keywords and forms the keyword set, denoted as $W = \{w_1, w_2, \cdots, w_m\}$. For each keyword $w_k \in W$, DO generates a binary vector of length $v$ bits, referred to as the index vector $v_{w_k}$. Initially, all elements of this index vector are set to 0. For each file $F_i$, a DO assigns a value of 1 to the $i$-$th$ bit in the index vector when it includes the keyword $w_k$, denoted by $v_{w_k}[i] = 1$. These individual index vectors $v_{w_k}$ collectively constitute the index vector set $V = \{v_{w_1}, v_{w_2}, \cdots, v_{w_m}\}$. For example, as shown in Fig. 2, suppose there are 10 files to be stored and 7 different keywords. Each file contains distinct keywords. The keyword $k_1$ is contained in files $F_1$, $F_3$, $F_7$ and $F_{10}$, the vector of keyword $w_1$ is $v_{w_1} = \{1,0,1,0,0,0,1,0,0,1\}$. The keyword $w_2$ is contained in files $F_2$, $F_6$ and $F_9$, the vector of keyword $v_{w_2} = \{0,1,0,0,0,1,0,0,1,0\}$. These index vectors form the index vector set $V = \{v_{w_1}, v_{w_2}, \cdots, v_{w_7}\}$.

**Algorithm 1 Index Generation**

**Require:** The secret key $x$, the keyword set $W$, the index vector set $V$

**Ensure:** The secure index $I$

```
1:  for Each $F_i \in F(1 \leqslant i \leqslant v)$ do
2:      Split $F_i$ into $n$ blocks $m_{i1}, m_{i2}, \cdots, m_{in}$
3:      for Each $w_k \in W(1 \leqslant k \leqslant m)$ do
4:          Obtain $v_{w_k}$ from $V$
5:          Compute $\pi_o(w_k)$
6:          Compute $ev_{\pi_o(w_k)} = v_{w_k} \oplus f_l(\pi_o(w_k))$
7:          Initiate an empty set $S_{w_k} = \emptyset$
8:          for Each $i \in [1, v]$ do
9:              if $v_{w_k}[i] == 1$ then
10:                 Insert $i$ to set $S_{w_k}$
11:             end if
12:         end for
13:         for Each $i \in S_{w_k}$ do
14:             for Each $1 \leqslant j \leqslant n$ do
15:                 Calculate
16:                 $\Omega_{w_k,ij} = [H_3(ID_i||j)^{-1} \times H_4(\pi_o(w_k)\|j)]^x$
17:             end for
18:         end for
19:     end for
20:     Set $\Omega_{\pi_o(w_k)} = \{\Omega_{w_k,i1}, \Omega_{w_k,i2}, \cdots, \Omega_{w_k,in}\}_{i, v_{w_k}[i]=1}$
21: end for
22: return $I = \{\pi_o(w_k), ev_{\pi_o(w_k)}, \Omega_{\pi_o(w_k)}\}_{k\in[1,m]}$
```

To guarantee the privacy of the file content and resist to guess the file content according to the keyword, a privacy method was designed. DO selects a pseudo random permutation key to blind the keyword by $\pi_o(w_k)$ for $k \in [1, m]$, the $\pi_o(w_k)$ is the address of each row in the index. Next, DO encrypts the index vector by $ev_{\pi_o(w_k)} = v_{w_k} \oplus f_l(\pi_o(w_k))$. To guarantee the accuracy in the auditing process, it is essential for the keyword to align with the index. The index plays a crucial role in the auditing procedure, and its design particulars are outlined in Alg. 1.

### B. The Construction

**1) $(params, msk, P_0) \leftarrow SysIni(\kappa)$**

With a security parameter $\kappa$, the *Key Generation Center* (KGC) generates the system's public parameters denoted as $params$=$(q,g,\mathbb{G},\mathbb{G}_T, e(\cdot,\cdot), H_1, H_2, H_3, H_4, \pi_{k_1}(\cdot), f_{k_2}(\cdot))$. In this configuration, $q$ represents a large prime and the order of multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$. The function $e(\cdot,\cdot)$: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ symbolizes a bilinear pairing. Furthermore, $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3, H_4 : \{0,1\}^* \rightarrow \mathbb{G}$ stand for four hash functions. $f_{key_1}(\cdot) : \{0,1\} \rightarrow [1,v]$, $\psi_{key2} : \{0,1\} \rightarrow [1,n]$ denotes pseudo-random permutation (PRP) with a random key $key_1$ and $key2$. $\pi_{key_3}(\cdot) : \{0,1\} \rightarrow \mathbb{Z}_q^*$ represents a pseudo-random function (PRF) with a random key $key_3$. The KGC completes the process by randomly selecting $\alpha \in \mathbb{Z}_q^*$ as the master secret key $msk$, and generating the master public key $P_0 = g^\alpha$.

**2) $(\{A_k = g^{a_k}\}_{k=1}^{s}, V) \leftarrow FileIni(\{F_v\})$**

- Suppose there are $v$ files to be stored. Firstly, DO splits each file into $n$ blocks and $s$ sectors. Each block is cut into $s$ sectors. Then, DO randomly picks $s$ values $a_k \in \mathbb{Z}_q^*$

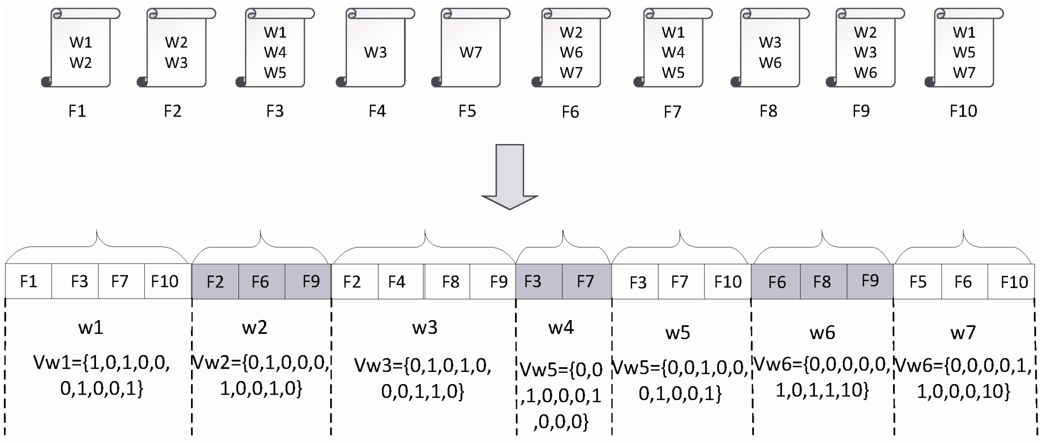

W1 W2
F1
W2 W3
F2
W1 W4 W5
F3
W3
F4
W7
F5
W2 W6 W7
F6
W1 W4 W5
F7
W3 W6
F8
W2 W3 W6
F9
W1 W5 W7
F10
F1 F3 F7 F10
F2 F6 F9
F2 F4 F8 F9
F3 F7
F3 F7 F10
F6 F8 F9
F5 F6 F10
w1
Vw1={1,0,1,0,0, 0,1,0,0,1}
w2
Vw2={0,1,0,0,0, 1,0,0,1,0}
w3
Vw3={0,1,0,1,0, 0,0,1,1,0}
w4
Vw5={0,0 ,1,0,0,0,1 ,0,0,0}
w5
Vw5={0,0,1,0,0, 0,1,0,0,1}
w6
Vw6={0,0,0,0,0, 1,0,1,1,10}
w7
Vw6={0,0,0,0,1, 1,0,0,0,10}

Fig. 2: An example of index extraction.

and keeps them secretly. DO generates $s$ public values $\{A_k = g^{a_k}\}_{k=1}^{s}$ and publishes them.

- DO begins by extracting a set of $m$ keywords, forming a keyword set denoted as $W = \{w_1, w_2, \cdots, w_m\}$. For each keyword $w_k \in W$, DO generates a binary vector of length $v$ bits to serve as the index vector $v_{w_k}$. Initially, all elements within this index vector are set to 0. Subsequently, for each file $F_i$, if it includes the keyword $w_k$, DO updates the $i$-$th$ bit of the index vector to 1: $v_{w_k}[i] = 1$. Collectively, these index vectors $v_{w_k}$ form the index vector set $V = \{v_{w_1}, v_{w_2}, \cdots, v_{w_m}\}$.

3) $\boldsymbol{(sk_{ID}, x, P_1) \leftarrow KeyGen(ID, msk)}$

The partial private key and secret key are generated as follows:

- DO initiates this process by sending its identity $ID$ to the KGC. Then the KGC generates the partial private key for the user's ID as follows: $sk_{ID} = H_1(ID)^{\alpha}$. This partial private key is subsequently sent to the user. When receiving the partial private key, DO performs a verification check. The verification involves confirming whether the equation $e(sk_{ID}, g) = e(H_1(ID), P_0)$ holds. If this validation step fails, DO rejects the key, and it is not accepted.
- DO proceeds by randomly selecting a value $x$ from the set $\mathbb{Z}_q^*$. This value $x$ is used as the secret key. The corresponding public key $P_1$ is generated as $P_1 = g^x$. Therefore, the private key of the user consists of two components: $(sk_{ID}, x)$, and the public key of the user is represented as $P_1$.

4) $\boldsymbol{I \leftarrow IndexGen(x, W, V)}$

- DO generates a vector $c_w = [n_{w_1}, n_{w_2}, \cdots, n_{w_m}]$, where $n_{w_i}, i \in [1, m]$ denotes the number of files containing the keyword.
- For each keyword $w_k$ in the keyword set $W$, DO calculates $\pi_o(w_k)$ as the address corresponding to each row in the index. Here, $o \in \mathbb{Z}_q^*$ represents the pseudo-random permutation key, which is selected by DO.
- For every keyword $w_k$ within the keyword set $W$, DO encrypts the index vector as $ev_{\pi_o(w_k)} = v_{w_k} \oplus f_l(\pi_o(w_k))$, where $l \in \mathbb{Z}_q^*$ is a pseudo-random function key selected by DO.

**Algorithm 2 Smart Contract-I**

```
Require: $Function\,name,\ invoked\,parameters$
Ensure: $Setting\,up\,functions$
 1: Structure: $Task$
 2: /* Define a structure of the auditing task */
 3: $taskID; fileName; numChallenges; startTime;$
    $endTime; chalStatus; proofStatus; recordList$
 4: /* the participant */
 5: $addressCS$
 6: Trapdoor information
 7: function: $newTask(user, name, n, CS)$
 8: /* Invoked by a user who has the auditing requirement */
 9: $task = tasks[taskID]$
10: $task.fileName = name$
11: $task.n = n$
12: $task.addressDO = msg.sender$
13: $task.addressCS = CS$
14: $task.num = 0$
15: $task.count = 0$
16: $task.trapdoorStatus = true$
17: $task.chalStatus = true$
18: $task.proofStatus = true$
19: function: $TrapdoorGen(\pi_o(w'), f_o(\pi_o)(w'), n'_w)$
20: $require(tasks[taskID].addressDO == msg.sender)$
21: $require(tasks[taskID].trapdoorStatus == true)$
22: $tasks[taskID].\pi_o(w') = \pi_o(w')$
23: $tasks[taskID].f_o(\pi_o)(w') = f_o(\pi_o)(w')$
24: $tasks[taskID].n'_w = n'_w$
25: $tasks[taskID].trapdoorStatus = false$
```

- For each keyword $w_k$ in the keyword set $W$, DO initializes an empty set $S_{w_k} = \emptyset$. Then, for each $i \in [1, v]$, if $v_{w_k}[i] = 1$, DO includes this file index $I$ in the set $S_{w_k}$. Assume that the number of elements that satisfy $v_{w_k}[i] = 1$ is denoted as $\xi$.
- For each keyword $w_k \in W$, DO calculates $\Omega_{\pi_o(w_k)} = \{\Omega_{w_k,i1}, \Omega_{w_k,i2}, \cdots, \Omega_{w_k,in}\}_{i, v_{w_k}[i]=1}$, where $\Omega_{w_k,ij} = [H_3(ID_i \| j)^{-1} \times H_4(\pi_o(w_k) \| j)]^x$, $j \in [1, n]$. DO sets $I = \{\pi_o(w_k), ev_{\pi_o(w_k)}, \Omega_{\pi_o(w_k)}\}_{k=1,2,\cdots,m}$.

5) $\boldsymbol{\Phi \leftarrow AuthGen(\{F_i\}_{i=1}^{v}, x, sk_{ID})}$

- Suppose the file set $F = \{F_1, F_2, \cdots, F_v\}$ and each file $F_i \in F$ has a unique identity $ID_i$. DO splits each file $F_i, i \in [1, v]$ into $n$ blocks, i.e., $F_i =$

**Algorithm 3** Smart Contract-II

```
Require: $Function : ChalGen(taskID, t_{start}, t_{end}, \triangle t, c)$
 1: $require(tasks[taskID].addressDO == msg.sender)$
 2: $require((t_{end} - t_{start})/(12 \times \triangle t) = 1)$
 3: $require(t_{end} > block.timestamp)$
 4: $require(tasks[taskID].chalStatus == true)$
 5: $tasks[taskID].startTime = t_{start}$
 6: $tasks[taskID].endTime = t_{end}$
 7: $tasks[taskID].numChallenges = c$
 8: $compute\, \mathcal{T} = \lfloor \frac{t_{end} - t_{start}}{12 \times \triangle t} \rfloor$
 9: $tasks[taskID].num = \mathcal{T}$
10: for $i \in [1, \mathcal{T}]$ do
11:     $recordList \leftarrow [t_{start} + (i-1) \times 12 \times \triangle t, t_{start} + i \times 12 \times \triangle t]$
12: end for
13: /* the retrieved blocks are $\mathcal{B}_i = \{B(t)\}$ */
14: $tasks[taskID].chalStatus = false$
15: function: $addProof(taskID, T^{(t)}, \{\mu_k^{(t)}\}, S'_w, result)$
16: $require(tasks[taskID].addressCS == msg.sender)$
17: $require(tasks[taskID].proofStatus == true)$
18: $tasks[taskID].numChallenges = c$
19: $tasks[taskID].auditresult = result$
20: $task.count + +$
21: if $(task.count > task.num)$ then
22:     $tasks[taskID].proofStatus = false$
23: end if
```

$\{m_{i1}, m_{i2}, \cdots, m_{in}\}$.

- DO blinded each block as a blind version, denoted as $b_{ij}$, using the formula $b_{ij} = m_{ij} + H_2(sk_{ID} \| ID_i \| i \| j)$, where $i \in [1, v]$ and $j \in [1, n]$. It's important to note that for each block $b_{ij}$, DO has the capability to recover the original plaintext $m_{ij}$ using the equation $m_{ij} = b_{ij} - H_2(sk_{ID} \| ID_i \| i \| j)$.
- Each block $b_{ij}$ is subdivided into $s$ sectors, represented as $\{b_{ijk}\}_{k \in [1,s]}$.
- For each blind data block $b_{ij}$, DO generates the authenticator $\sigma_{ij} = sk_{ID} \times [H_3(ID_i \| j) \times g^{\sum_{k=1}^{s} a_k b_{ijk}}]^x$.
- DO sets $\Phi = \{\sigma_{ij}\}, (1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s)$.

Finally, DO uploads the data $F = \{F_1, F_2, \cdots, F_v\}$, the authenticator $\Phi = \{\sigma_{ij}\}, (1 \leqslant i \leqslant n, 1 \leqslant j \leqslant s)$ to the cloud and the index structure $I = \{\pi_o(w_k), ev_{\pi_o(w_k)}, \Omega_{\pi_o(w_k)}\}_{k=1,2,\cdots,m}$ to the smart contract.

6) $\boldsymbol{(T_{w'}, n_{w'}) \leftarrow TrapdoorGen(w')}$
DO generates the search trapdoor as $T_{w'} = \{\pi_o(w'), f_l(\pi_o(w'))\}$ based on keyword $w'$. DO retrieves $n_{w'}$ from $c_w$, and sends the trapdoor $T_{w'}$ and $n_{w'}$ to the smart contract.

Smart contracts are designed to eliminate centralized intermediaries and facilitate transparent auditing processes, leveraging four modular functions to ensure trustless execution and verifiability. Assume that the entity DO and CS have registed an address in blockchain. The DO and CS have a consensus agreement on auditing the data. After that, the DO initiates a smart contract, as shown in Alg. 2, into BC. In the smart contract, the DO defines the auditing task, including the file name, the number of challenge blocks, the address of DO and CS in the blockchain.

7) $\boldsymbol{Chal \leftarrow ChalGen(T_{w'})}$ DO initiates an auditing process through a smart contract, establishing an agreement denoted as $Ag$ with five key elements: $Ag = \{t_{start}, t_{end}, \triangle t, c\}$. In this context: $t_{start}$ represents the starting timestamp, which is determined by DO and signifies the future point in time when the auditing process commences. $t_{end}$ is the endpoint of the auditing process, indicating when the audit concludes. $\triangle t$ corresponds to the average time required to generate a single block within recent time periods. $\mathcal{T}$ stands for the number of times the auditing service will be checked or queried. It is calculated as $\mathcal{T} = \lceil \frac{t_{end} - t_{start}}{12 \times \triangle t} \rceil$, ensuring that it is equal to or greater than 1. Let $B(t)$ represent the function that calculates the block number containing the timestamp $t$. For each auditing task, CS will wait about $12 \times \triangle t$ time before generating the proof information. The challenge information will be retrieved from a succession of blocks. That is, for each auditing task $i$, $i \in [1, \mathcal{T}]$, $\mathcal{B}_i = \{B(t)\}$, $t \in [t_{start} + (i-1) \times 12 \times \triangle t, t_{start} + i \times 12 \times \triangle t]$. $c$ is the number of challenge blocks.

8) $\boldsymbol{(Proof, S_{w'}) \leftarrow ProofGen(Chal, I, \Phi)}$
CS parses the challenge information from BC. For any auditing task $t$, $t \in [1, \mathcal{T}]$:

- CS generates $k_{t1} = H_2(1 \| \mathcal{B}_t)$, $k_{t2} = H_2(2 \| \mathcal{B}_t)$. Then, CS generates the challenge information $\{C = \{(j_t, v_{ij_t})\}\}_{j_t \in [1,c], i \in [1,v]}\}$, where $j_t = \psi_{k_{t1}}(\beta)$ for $\beta \in [1, c]$ and $v_{ij_t} = \pi_{x_{i_t}}(\beta)$ for $x_{i_t} = \pi_{k_{t2}}(i)$, $i \in [1, v]$ and $\beta \in [1, c]$. Suppose $Q_t = \{j_t = \psi_{k_{t1}}(\beta)\}_{\beta \in [1,c]}$.
- Based on the address $\pi_o(w_k) = \pi_o(w')$, CS locates the associated encrypted row $ev_{\pi_o(w_k)}$ and $\Omega_{\pi(w_k)}$ in the index. Subsequently, CS decrypts the corresponding encrypted index vector, represented as $v_{w_k} = ev_{\pi_o(w_k)} \oplus f_l(\pi_o(w_k))$.
- The cloud initializes an empty set $S_{w'} = \emptyset$. For each $i \in [1, n]$, if $v_{w'}[i] = 1$, CS includes $i$ in the set $S_{w'}$.
- CS calculates $T^{(t)} = \prod_{i \in S_{w'}} \prod_{j_t \in Q_t} (\sigma_{ij_t})^{v_{ij_t}}$, $\mu_k^{(t)} = \sum_{i \in S_{w'}} \sum_{j_t \in Q_t} (v_{ij_t} \times b_{ij_t k})$.
  CS sets the auditing proof $Proof^{(t)} = \{T^{(t)}, \{\mu_k^{(t)}\}_{1 \leqslant k \leqslant s}\}$.

9) $\boldsymbol{(true/false) \leftarrow SelfVerify(Chal, Proof)}$
For each auditing task, after obtaining the proof, CS checks the integrity by Eq. (1). CS sends the proof information $Proof = \{T^{(t)}, \{\mu_k^{(t)}\}_{1 \leqslant k \leqslant s}\}$, the set $S_{w'}$ and the audit result $true/false$ to the smart contract, as shown in Alg. 3.

$$\begin{aligned}
&e(T^{(t)} \times \prod_{i \in S_{w'}} \prod_{j_t \in Q} (\Omega_{w', ij_t})^{v_{ij_t}}, g) \\
&= e(\prod_{j_t \in Q} (H_4(\pi_o(w_k) \| j_t)^{\sum_{i \in S_{w'}} v_{ij_t}}) \times \prod_{k=1}^{s} A_k^{\mu_k^{(t)}}, R) \quad (1) \\
&\times e((H_1(ID))^{\sum_{i \in S_{w'}} \sum_{j_t \in Q} v_{ij_t}}, P_0).
\end{aligned}$$

## C. Arbitration

- **Compensation**. When DO obtains the auditing result $false$ from blockchain, DO can charge for economic losses due to data corruption.
- **Dispute Arbitration**. When the following disputes happen: 1) The data loss happen but CS does not generate correct report on the blockchain; 2) The data is stored correctly but DO charges for economic losses due to data

corruption. An arbitrator can deal with the disputing as follows:
For each auditing task $t$, $t \in [1, \mathcal{T}]$, the arbitrator retrieves the challenge information from blockchain, then the arbitrator generates $\widetilde{k_{t1}} = H_2(1\|\mathcal{B}_i)$ and $\widetilde{k_{t2}} = H_2(2\|\mathcal{B}_i)$. The arbitrator generates the challenge information $\{\widetilde{C} = \{(\widetilde{j_t}, \widetilde{v_{j_t}})\}_{\widetilde{j_t}\in[1,c], i\in[1,v]}\}$, where $\widetilde{j_t} = \psi_{\widetilde{k_{t_1}}}(\beta)$ for $\beta \in [1, c]$, and $\widetilde{v_{ij_t}} = \pi_{\widetilde{x_{i_t}}}(\beta)$ for $\widetilde{x_{i_t}} = \pi_{\widetilde{k_{t_2}}}(i)$, $i \in [1, v]$ and $\beta \in [1, c]$. Suppose $Q_t = \{\widetilde{j_t} = \psi_{k_{t1}}(\beta)\}_{\beta\in[1,c]}$.
The arbitrator checks the validity of the verification according to the equation

$$\begin{aligned} &e(T^{(t)} \times \prod_{i\in S_{w'}} \prod_{\widetilde{j_t}\in Q_t} (\Omega_{w', i\widetilde{j_t}})^{\widetilde{v_{ij_t}}}, g) \\ &= e((\prod_{\widetilde{j_t}\in Q_t} (H_4(\pi_o(w_k)\|\widetilde{j_t})^{\sum_{i\in S_{w'}} \widetilde{v_{ij_t}}})) \times \prod_{k=1}^{s} A_k^{\mu_k^{(t)}}, R) \\ &\times e((H_1(ID))^{\sum_{i\in S_{w'}} \sum_{\widetilde{j_t}\in Q_t} \widetilde{v_{ij_t}}}, P_0). \end{aligned} \tag{2}$$

If the Eq. (2) holds, it means that CS is honest and the files are stored correctly. Otherwise, it means that the data was lost. CS should pay for DO's losses.

## VI. Security Analysis

We provide the security analysis of the proposed scheme, including the unforgeability of the authenticators and the unforgeability of the proof information.

**Theorem 1.** (Unforgeability of authenticator). *There does not exist a PPT adversary who forges a valid authenticator with non-negligible probability under the CDH assumption.*

*Proof.* The security proof is provided in Appendix. □

**Theorem 2.** (Searchable-auditing soundness). *There does not exist a PPT adversary who forges an valid proof information with non-negligible probability under the DL and CDH assumption.*

*Proof.* $Game\,0$: This game is the same as defined in security model.

$Game\,1$: This game is similar to the game in $Game\,0$, with a little difference that the the proof $T^*$ does not appear in the record list, the simulator $\mathcal{S}$ aborts. Given the valid proof $Proof^{(t)} = \{T^{(t)}, \{\mu_k^{(t)}\}_{1\leqslant k\leqslant s}\}$, it satisfies Eq. 1,

$$\begin{aligned} &e(T^{(t)}, g) = e((\prod_{j_t\in Q} (H_4(\pi_o(w_k)\|j_t)^{\sum_{i\in S_{w'}} x_{i_t}})^{v_{j_t}}) \times \\ &\prod_{k=1}^{s} A_k^{\mu_k^{(t)}}, R) \times e((H_1(ID))^{\sum_{i\in S_{w'}} x_{i_t} \sum_{j\in Q} v_{j_t}}, P_0). \end{aligned}$$

If a forged proof $Proof^{(t)*} = \{T^{(t)*}, \{\mu_k^{(t)*}\}_{1\leqslant k\leqslant s}\}$ passes the verification, it also satisfies

$$\begin{aligned} &e(T^{(t)*}, g) = e((\prod_{j_t\in Q} (H_4(\pi_o(w_k)\|j_t)^{\sum_{i\in S_{w'}} x_{i_t}})^{v_{j_t}}) \times \\ &\prod_{k=1}^{s} A_k^{\mu_k^{(t)*}}, R) \times e((H_1(ID))^{\sum_{i\in S_{w'}} x_{i_t} \sum_{j\in Q} v_{j_t}}, P_0). \end{aligned} \tag{3}$$

Since the $Game\,1$ aborts, it satisfies $\prod_{k=1}^{s} A_k^{\mu_k} \neq \prod_{k=1}^{s} A_k^{\mu_k^*}$, thus there exists $\mu_k \neq \mu_k^*$ for $\in [1, s]$. Let $\triangle\mu_k = \mu^* - \mu$. If $Game\,1$ aborts, $\mathcal{S}$ can solve the CDH problem.

Given $(g, g^a, g^b)$, $\mathcal{S}$ aims to output $g^{ab}$. For $j \in [1, k]$, $\mathcal{S}$ chooses $\alpha_j, \beta_j \in \mathbb{Z}_q^*$ and sets $A_k = g^{\alpha_j} g^{b\beta_j}$. For $H_1(ID)$ query, $\mathcal{S}$ sets $H_1(ID) = g^{\alpha}$. For $H_3(ID_i\|j)$, $\mathcal{S}$ sets $H_3(ID_i\|j) = \frac{g^{r_{ij}}}{\prod_{k=1}^{s}(g^{\alpha_k}\times g^{b\beta_k})^{b_{ijk}}}$. For $H_4(\pi_o(w_k)\|j)$, $\mathcal{S}$ sets $H_4(\pi_o(w_k)\|j) = g^{r_{kj}}$. Thus, the simulator can obtain:

$$\begin{aligned} \sigma_{ij} &= sk_{ID} \times [H_3(ID_i\|j) \times g^{\sum_{k=1}^{s} a_k b_{ijk}}]^x \\ &= g^{\alpha\gamma} \times (\frac{g^{r_{ij}}}{\prod_{k=1}^{s}(g^{\alpha_k} \times g^{b\beta_k})^{b_{ijk}}} \times \prod_{k=1}^{s} (g^{\alpha_k \times g^{b\beta_k}})^{b_{ijk}})^a \\ &= g^{\alpha\gamma} \times (g^{r_{ij}})^a. \end{aligned}$$

The index can be obtained

$$\begin{aligned} \Omega_{\pi_o(w_k)} &= [H_3(ID_i\|j)^{-1} \times H_4(\pi_o(w_k)\|j)]^x \\ &= ((\frac{g^{r_{ij}}}{\prod_{k=1}^{s}(g^{\alpha_k\times g^{b\beta_k}})^{b_{ijk}}})^{-1} \times g^{r_{kj}})^a. \end{aligned}$$

We obtain the following via dividing the Eq. (3) by Eq. (1), $e(\frac{T^*}{T}, g) = e(\prod_{k=1}^{s} A_k^{\triangle\mu_k}, g^a)$. Thus, $\frac{T^*}{T} = (g^{\sum_{k=1}^{s}\alpha_k} \times g^{b\times\sum_{k=1}^{s}\beta_k})^{a\times\triangle\mu_k}$. The value $g^{ab} = (\frac{T^*}{T} \times g^{-\sum_{k=1}^{s}\alpha_k\times a\times\triangle\mu_k})^{\frac{1}{\sum_{k=1}^{s}\beta_k\times\triangle\mu_k}}$. Since $\triangle\mu_k \neq 0$, the probability $Pr(\sum_{k=1}^{s}\beta_k \times \triangle\mu_k \neq 0) = 1 - \frac{1}{q^s}$, which is a non-negligible probability and disobey the CDH assumption.

$Game\,2$: This game is as same as $Game\,1$, with a little difference that the forged proof $\mu^*$ does not appear in the record list, $\mathcal{S}$ aborts. Since the valid auditing proof $Proof^{(t)} = \{T^{(t)}, \{\mu_k^{(t)}\}_{1\leqslant k\leqslant s}\}$ and the forged proof $Proof^{(t)*} = \{T^{(t)}, \{\mu_k^{(t)*}\}_{1\leqslant k\leqslant s}\}$ all pass the verification, it can be obtained that

$$\begin{aligned} &e(T^{(t)}, g) = e((\prod_{j_t\in Q} (H_4(\pi_o(w_k)\|j_t)^{\sum_{i\in S_{w'}} x_{i_t}})^{v_{j_t}}) \times \\ &\prod_{k=1}^{s} A_k^{\mu_k^{(t)}}, R) \times e((H_1(ID))^{\sum_{i\in S_{w'}} x_{i_t} \sum_{j\in Q} v_{j_t}}, P_0). \end{aligned}$$

and

$$\begin{aligned} &e(T^{(t)}, g) = e((\prod_{j_t\in Q} (H_4(\pi_o(w_k)\|j_t)^{\sum_{i\in S_{w'}} x_{i_t}})^{v_{j_t}}) \times \\ &\prod_{k=1}^{s} A_k^{\mu_k^{(t)*}}, R) \times e((H_1(ID))^{\sum_{i\in S_{w'}} x_{i_t} \sum_{j\in Q} v_{j_t}}, P_0). \end{aligned}$$

If $\mathcal{S}$ aborts, the DL problem is solved by the simulator.

Given $(g, g^b)$, $\mathcal{S}$ aims to retrieve $b$. For $k \in [1, s]$, $\mathcal{S}$ chooses $\alpha_k, \beta_k \in \mathbb{Z}_q^*$ and sets $A_k = g^{\alpha_k} g^{b\beta_k}$. Since $\mathcal{S}$ aborts, $\mu_k \neq \mu_k^*$. Let $\triangle\mu_k = \mu_k^* - \mu_k$. From $Game\,1$, we can obtain $T^* = T$. Thus, $\prod_{k=1}^{s} A_k^{\mu_k} = \prod_{k=1}^{s} A_k^{\mu_k^*}$. That is,

$$\begin{aligned} &(g^{\sum_{k=1}^{s}\alpha_k} \times g^{b\sum_{k=1}^{s}\beta_k})^{\mu_k} = (g^{\sum_{k=1}^{s}\alpha_k} \times g^{b\sum_{k=1}^{s}\beta_k})^{\mu_k^*} \\ &\Rightarrow (g^{\sum_{k=1}^{s}\alpha_k} \times g^{b\sum_{k=1}^{s}\beta_k})^{\triangle\mu} = 1 \\ &\Rightarrow g^{-\sum_{k=1}^{s}\alpha_k\triangle\mu_k} = g^{\sum_{k=1}^{s}\beta_k\triangle\mu_k} \\ &\Rightarrow a = \frac{-\sum_{k=1}^{s}\alpha_k}{\sum_{k=1}^{s}\beta_k}. \end{aligned}$$

As long as $\sum_{k=1}^{s}\beta_k \neq 0$, the DL problem is solved. Since $\sum_{k=1}^{s}\beta_k \neq 0$, the probability $Pr(\sum_{k=1}^{s}\beta_k \neq 0) = 1 - \frac{1}{q^s}$, which is a non-negligible probability and disobey the DL assumption. □

**Theorem 3.** (Trapdoor and index indistinguishability) *If PRP and PRF are secure, the proposed scheme can achieve trapdoor and index indistinguishability.*

*Proof.* **Phase 1**: $Index\, Query$: $\mathcal{A}_4$ queries the index of keyword $w'$. $\mathcal{C}$ randomly sets $\pi_o(w')$ and $f_l(\pi_o(w'))$. Then $\mathcal{C}$ retrievs $ev_{\pi_o(w')} = v_{w'} \oplus f_l(\pi_o(w'))$. According to $w'$, $\mathcal{C}$ generates $\Omega_{\pi_o(w')} = \{\Omega_{\pi_o(w'),i1}, \Omega_{\pi_o(w'),i2}, \cdots, \Omega_{\pi_o(w'),in}\}_{i,v_{w'}[i]=1}$, where $\Omega_{\pi_o(w'),ij} = [H_3(ID_i\|j)^{-1} \times H_4(\pi_o(w'\|j))]^x$, $j \in [1,n]$. DO sets $I' = \{\pi_o(w'), ev_{\pi_o(w')}, \Omega_{\pi_o}(w')\}_{k=1,2,\cdots,m}$.

$Trapdoor\, Query$: $\mathcal{C}$ queries the trapdoor of keyword $w'$. If this query has been queried before, $\mathcal{C}$ returns the corresponding $T_{w'}$. Otherwise, $\mathcal{C}$ generates $\pi_o(w')$ and $f_l(\pi_o(w'))$. Finally, $\mathcal{C}$ returns $T_{w'} = \{\pi_o(w'), f_l(\pi_o(w'))\}$ to $\mathcal{A}_4$.

**Challenge**: $\mathcal{A}_4$ chooses two keywords $\{w_0^*, w_1^*\}$ and sends it to $\mathcal{C}$. $\mathcal{C}$ randomly flips a coin, designating heads (heads-up) as 1 and tails (tails-down) as 0. Using the outcome of the coin toss, $\mathcal{C}$ executs the trapdoor generation algorithm to produce the trapdoor $T_{w_\beta} = \{\pi_o(w_\beta), f_l(\pi_o(w_\beta))\}$. $\mathcal{C}$ executes the index generation algorithm to produce the index $I_\beta = \{\pi_o(w_\beta), ev_{\pi_o(w_\beta)}, \Omega_{\pi_o}(w_\beta)\}_{k=1,2,\cdots,m}$. Finally, $\mathcal{C}$ sends the index $I_\beta$ and the trapdoor $T_{w_\beta}$ to $\mathcal{A}_4$.

**Phase 2**: $Index\, Query$: $\mathcal{A}_4$ queries the index of keyword $w'(w' \notin \{w_0^*, w_w^*\})$. If this query has been queried before, $\mathcal{C}$ returns the corresponding $I'$. $\mathcal{C}$ randomly sets $\pi_o(w')$ and $f_l(\pi_o(w'))$. Then $\mathcal{C}$ retrievs $ev_{\pi_o(w')} = v_{w'} \oplus f_l(\pi_o(w'))$. According to $w'$, $\mathcal{C}$ generates $\Omega_{\pi_o(w')} = \{\Omega_{\pi_o(w'),i1}, \Omega_{\pi_o(w'),i2}, \cdots, \Omega_{\pi_o(w'),in}\}_{i,v_{w'}[i]=1}$, where $\Omega_{\pi_o(w'),ij} = [H_3(ID_i\|j)^{-1} \times H_4(\pi_o(w'\|j))]^x$, $j \in [1,n]$. DO sets $I' = \{\pi_o(w'), ev_{\pi_o(w')}, \Omega_{\pi_o}(w')\}_{k=1,2,\cdots,m}$.

$Trapdoor\, Query$: $\mathcal{C}$ queries the trapdoor of keyword $w'(w' \notin \{w_0^*, w_w^*\})$. If this query has been queried before, $\mathcal{C}$ returns the corresponding $T_{w'}$. Otherwise, $\mathcal{C}$ generates $\pi_o(w')$ and $f_l(\pi_o(w'))$. Finally, $\mathcal{C}$ returns $T_{w'} = \{\pi_o(w'), f_l(\pi_o(w'))\}$ to $\mathcal{A}_4$.

Due to the security of PRP and PRF, $(I_0, I_1)$ and $(T_{w_0}, T_{w_1})$ are indistinguishable from each other. □

**Theorem 4.** (Transparent-auditing). *The auditing process is transparent.*

*Proof.* During the auditing process, all pertinent information is uploaded onto the blockchain via smart contracts. Specifically, for index information, DO uploads the index structure $I = \{\pi_o(w_k), ev_{\pi_o(w_k)}, \Omega_{\pi_o(w_k)}\}_{k=1,2,\cdots,m}$ to the smart contract. Regarding trapdoor information, DO transmits the trapdoor $T_w'$ and $n_w'$ to the smart contract. The challenge information is generated leveraging data stored on the blockchain and smart contracts. As for proof information, CS forwards the proof data $Proof = \{T^{(t)}, \{\mu_k^{(t)}\}_{1 \leqslant k \leqslant s}\}$, the set $S_{w'}$ and the audit result $true/false$ to the smart contract. With the inherent transparency of blockchain technology, all content stored on the blockchain remains traceable and immutable. By tracing the entirety of the information stored on the blockchain, the verification process remains observable. Consequently, the auditing process is rendered transparent. □

TABLE II: Comparison of Functionality

| Scheme | [28] | [39] | [17] | [40] | Ours |
|---|---|---|---|---|---|
| Type | II | III | I | II | III |
| $P_1$ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $P_2$ | × | × | × | ✓ | ✓ |
| $P_3$ | × | × | ✓ | × | ✓ |
| $P_4$ | × | × | × | ✓ | ✓ |

I: Public Key Infrastructure; II: Identity-based Cryptography; III: Certificateless-based Cryptography.
$P_1$: Public auditing; $P_2$: Without relying on trust TPA; $P_3$: Keyword searchable auditing; $P_4$: Traceability.

**Theorem 5.** (Multitask). *DO does not have to keep online all time. The proposed scheme achieves multiple challenge tasks in one time.*

*Proof.* In the proposed scheme, multiple audit task challenge contents are generated a time in a decentralized manner by DO with the assistance of smart contracts. Given a future start timestamp $t_{start}$ and an end timestamp $t_{end}$, the challenge information is determined by $\mathcal{T} = \lceil \frac{t_{end} - t_{start}}{12 \times \triangle t} \rceil$. Each challenge is associated with a series of data blocks, denoted by $\mathcal{B}_i = \{B(t)\}$, $t \in [t_{start} + (i-1) \times 12 \times \triangle t, t_{start} + i \times 12 \times \triangle t]$, for $i \in [1, \mathcal{T}]$. This approach reduces the frequency of challenges presented to TPA, thus relieving DO from the obligation of maintaining an online presence continuously. When initiating an auditing challenge, CS must await the generation of an auditing challenge message. In summary, CS conducts the auditing tasks sequentially based on the challenge information generated from the blockchain and smart contracts. Thus ,the proposed scheme facilitates the execution of multiple challenge tasks simultaneously. □

**Theorem 6.** (Arbitration). *A fair arbitration is achieved when dispute happens between DO and CS.*

*Proof.* There are two scenarios that may precipitate conflict between DO and CS. The first arises when data loss occurs but CS fails to generate accurate reports on the blockchain. The second arises when data is accurately stored, yet DO incurs economic losses due to data corruption. To ensure equitable resolution of such disputes, an arbitrator is introduced. All information stored on the blockchain is easily traceable and accessible. The results of each audit can be re-verified by the arbitrator. In the event that CS is found to have provided erroneous reports, it is incumbent upon them to compensate DO for financial losses. □

## VII. Performance Evaluation

### A. Theoretical Analysis

*1) Functionality Comparison:* We firstly list a functionality comparison with some related work in recent years in TABLE II. We compare the functionality with scheme [17], [28], [39], [40] including authenticator type, public auditing, reliance on a trusted TPA, keyword searchable auditing and traceability. We

TABLE III: Computation Overhead

| Phase | AuthGen | IndexGen | ProofGen | Verify |
|---|---|---|---|---|
| [28] | $(2+s)\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ | — | $c\times Exp_{\mathbb{G}} + c\times Mul_{\mathbb{Z}_q^*}$ | $3\times Pair + (1+c+2)\times Exp_{\mathbb{G}} + (c+s-2)\times Mul_{\mathbb{G}}$ |
| [39] | $(2+s)\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ | — | $c\times Exp_{\mathbb{G}} + c\times Mul_{\mathbb{Z}_q^*}$ | $3\times Pair + (1+c+2)\times Exp_{\mathbb{G}} + (c+s-2)\times Mul_{\mathbb{G}}$ |
| [17] | $H_{\mathbb{G}} + Mul_{\mathbb{G}} + Exp_{\mathbb{G}}$ | $m\times s\times[(\lvert S_w\rvert+2)\times H_{\mathbb{G}} + (\lvert S_w\rvert+1)\times Mul_{\mathbb{G}} + Exp_{\mathbb{G}}]$ | $2cExp_{\mathbb{G}} + (\lvert S_w\rvert \times c + c)Mul_{\mathbb{G}} + \lvert S_w\rvert \times cMul_{\mathbb{Z}_q^*}$ | $2\times Pair + 2\times c\times H_{\mathbb{G}} + (c+1)\times Mul_{\mathbb{G}} + (c+1)\times Exp_{\mathbb{G}}$ |
| [40] | $(2+n)\times Exp_{\mathbb{G}} + Mul_{\mathbb{Z}_q^*} + Mul_{\mathbb{G}}$ | — | $c\times Exp_{\mathbb{G}} + c\times Mul_{\mathbb{Z}_q^*}$ | $2\times Pair + (c+1)\dot{E}xp_{\mathbb{G}} + c\times Mul_{\mathbb{G}}$ |
| Our scheme | $2\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ | $m\times s\times\lvert S_w\rvert\times 2\times H_{\mathbb{G}} + (\lvert S_w\rvert)\times Mul_{\mathbb{G}} + Pow_{\mathbb{G}}$ | $(\lvert S_w\rvert + c)\times Exp_{\mathbb{G}} + (\lvert S_w\rvert + c)\times Mul_{\mathbb{G}} + \lvert S_w\rvert\times c\times Mul_{\mathbb{Z}_q^*}$ | $3\times Pair + \lvert S_w\rvert\times c\times H_{\mathbb{G}} + (\lvert S_w\rvert\times c + s - 2)\times Mul_{\mathbb{Z}_q^*} + (\lvert S_w\rvert\times c + s + 1)\times Exp_{\mathbb{G}} + Mul_{\mathbb{Z}_q^*}$ |

TABLE IV: Communication Overhead

| Phase | AuthGen | IndexGen | ChalGen | ProofGen |
|---|---|---|---|---|
| [28] | $\lvert F\rvert + (s+n+2)\lvert\mathbb{G}\rvert$ | — | $\lvert n\rvert + 2\lvert\mathbb{Z}_q^*\rvert$ | $(3+s)\lvert\mathbb{G}\rvert + s\lvert\mathbb{Z}_q^*\rvert$ |
| [39] | $\lvert F\rvert + (n+1)\lvert\mathbb{G}\rvert$ | — | $(c+1)\lvert n\rvert + c\lvert\mathbb{Z}_q^*\rvert$ | $2\lvert\mathbb{G}\rvert + (s+1)\lvert\mathbb{Z}_q^*\rvert$ |
| [17] | $\lvert F\rvert + n\lvert\mathbb{G}\rvert$ | $2m\lvert n\rvert$ | $3\lvert\mathbb{Z}_q^*\rvert + 2\lvert n\rvert$ | $\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_q^*\rvert$ |
| [40] | $\lvert F\rvert + (n+1)\lvert\mathbb{G}\rvert + 3\lvert n\rvert$ | — | $\lvert n\rvert + 2\lvert\mathbb{Z}_q^*\rvert$ | $\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_q^*\rvert$ |
| Our scheme | $\lvert F\rvert + n\lvert\mathbb{G}\rvert$ | $2m\lvert n\rvert$ | $5\lvert n\rvert$ | $\lvert\mathbb{G}\rvert + s\lvert\mathbb{Z}_q^*\rvert$ |

also list the cryptosystem in these schemes. From the TABLE, scheme [28] and [40] are all base on identity cryptography, scheme [39] and our scheme rely on certificateless cryptography, while scheme [17] is based on a public key infrastructure. In terms of functionality, all schemes support public auditing. Both scheme [40] and our scheme do not rely on a trusted TPA and support traceability. Both scheme [17] and our scheme support keyword searchable auditing. To sum up, our scheme has advantages in without relying on a trusted TPA, keyword searchable auditing and traceability.

*2) Computation Costs:* In the analysis of computational costs, our focus is on particular operations, which include bilinear pairing operations, exponentiation operations on $\mathbb{G}$, and multiplication operations on $\mathbb{Z}_q^*$ and $\mathbb{G}$. The comparison is presented in terms of four main aspects: **AuthGen**, **IndexGen**, **ProofGen** and **Verify** in TABLE III.

For **AuthGen** algorithm, schemes [28] and scheme [39] have similar computation costs as $(2+s)\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ and $(2+s)\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ respectively. Scheme [40] has smaller computation costs as $H_{\mathbb{G}} + Mul_{\mathbb{G}} + Exp_{\mathbb{G}}$ due to its PKI system. Compared to schemes [28] and [39], our scheme has the smallest computation costs as $2\times Exp_{\mathbb{G}} + s\times Mul_{\mathbb{Z}_q^*} + 2\times Mul_{\mathbb{G}}$ because our scheme does not need to generate a sequence of public parameters like $\{A_k\}_{k=1}^{s}$. For **IndexGen** algorithm, only scheme [17] and our scheme support searchable auditing. The computation costs for scheme [17] and our scheme are $m\times s\times[(\lvert S_w\rvert+2)\times H_{\mathbb{G}} + (\lvert S_w\rvert+1)\times Mul_{\mathbb{G}} + Exp_{\mathbb{G}}]$ and $m\times s\times\lvert S_w\rvert\times 2\times H_{\mathbb{G}} + (\lvert S_w\rvert)\times Mul_{\mathbb{G}} + Pow_{\mathbb{G}}$ respectively. Compared to scheme [17], our scheme has smaller computation costs because our scheme does not require a block position hash operation for each block but still achieves index matching. For **ProofGen** algorithm, scheme [28], [39] and [40] all have the same computation costs as $c\times Exp_{\mathbb{G}} + c\times Mul_{\mathbb{Z}_q^*}$. Compared to scheme [17], our scheme has higher computation costs, as it incurs $c\times Exp_{\mathbb{G}} + c\times Mul_{\mathbb{Z}_q^*}$ due to its resistance to same keyword summation attacks. For **Verify** algorithm, scheme [28] and [39] have the same computation costs, which are $3\times Pair + (1+c+2)\times Exp_{\mathbb{G}} + (c+s-2)\times Mul_{\mathbb{G}}$. Scheme [39] and [40] have similar computation costs as $2\times Pair + 2\times c\times H_{\mathbb{G}} + (c+1)\times Mul_{\mathbb{G}} + (c+1)\times Exp_{\mathbb{G}}$ and $2\times Pair + (c+1)\dot{E}xp_{\mathbb{G}} + c\times Mul_{\mathbb{G}}$ respectively, due to their use of the same cryptography system. The computation costs of our scheme are related to the number of files containing the same keyword and are $3\times Pair + \lvert S_w\rvert\times c\times H_{\mathbb{G}} + (\lvert S_w\rvert\times c + s - 2)\times Mul_{\mathbb{Z}_q^*} + (\lvert S_w\rvert\times c + s + 1)\times Exp_{\mathbb{G}} + Mul_{\mathbb{Z}_q^*}$.

*3) Communication Costs:* For communication costs, we mainly consider the communication cost in terms of $\lvert\mathbb{G}\rvert$, $\lvert\mathbb{Z}_q^*\rvert$ and $\lvert n\rvert$. We compare the communication costs for the **AuthGen**, **IndexGen**, **ChalGen**, and **ProofGen** algorithms in TABLE IV.

For the **AuthGen** algorithm, except for scheme [28], which needs to transmit $s$ group elements $s\lvert\mathbb{G}\rvert$, the communication costs of the other schemes have a small difference. For the **IndexGen** algorithm, the communication content mainly includes the index table. Scheme [17] and our scheme have the same communication costs, which are $2m|n|$. For the **ChalGen** algorithm, both scheme [28] and [40] transmit the challenge numbers $|n|$ and two challenge seeds $2|\mathbb{Z}_q^*|$ to the CS, while scheme [17] should transmit extra trapdoor information $|n| + |\mathbb{Z}_q^*|$. Compared to other schemes, scheme [40] has the highest communication overhead, amounting to $(c+1)|n|+c|\mathbb{Z}_q^*|$, since it needs to transmit the whole challenge set. Our scheme adopts a decentralized way of challenge information generation, requiring only $5|n|$ communication costs. For the **ProofGen** algorithm, the communication costs are related to the number of sectors in scheme [28], [39] and our scheme, which are $(3+s)|\mathbb{G}| + s|\mathbb{Z}_q^*|$, $(3+s)|\mathbb{G}| + s|\mathbb{Z}_q^*|$ and $|\mathbb{G}| + s|\mathbb{Z}_q^*|$, respectively.

## *B. Performance Evaluation*

This section describes the simulation experiments. All experiments were conducted on a system running Windows 10 with an Intel i7 2.5 GHz CPU and 8 GB of memory. We implemented the scheme using the Java programming language and the *Pairing Based Cryptography* (PBC) library. In our evaluation, we utilized type pairings constructed on the curve defined by the equation $y^2 = x^3 + x$ over the finite field $\mathcal{F}_q$, where $q = 3\,mod\,4$. To assess the performance of our proposed scheme, we conducted a comparative analysis of its computational costs in comparison to related work as presented in [8], [17], [28], [39].

**—Evaluation time cost for authenticator generation**.

Fig. 3 illustrated the relationship of the time computation costs of authenticator generation and the block number of the original file. The block number was varied from 1000 to 10000 with a step size of 500, we observed that the time costs of the authenticator generation increased as the block number of original file grew. Our scheme outperformed other schemes such as [28], [39] in terms of efficiency. In Fig. 4, we examined the relationship of the time computation costs of authenticator generation and file size. The file size was set to 5 MB and 10 MB respectively. We observed that the computation costs decreased as the number of block segments increased. Additionally, our scheme exhibited greater efficiency compared to the scheme in [28], [39]. For example, when the number of blocks was configured to 4500, the time required for authenticator generation in schemes [28], [39] and our proposed approach was 143.533, 146.338, and 138.798 seconds, respectively. Compared to scheme [39], the time cost was reduced by approximately 5%. This reduction in time cost was achieved because certain public values could be pre-computed in advance. Additionally, it was important to note that the computation costs of authenticator generation increased with the file size, as expected.

**—Evaluation time cost for index generation**.

Fig. 5 illustrated the relationship of index generation time and the number of keywords and separated data block numbers in a single block. The number of separated data block numbers was set from 100 to 1000 with step size 100. In case of resisting the same keyword files summation attack,

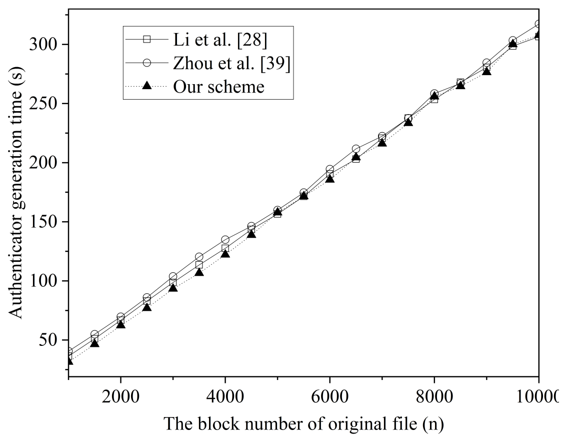


Fig. 3: The authenticator generation time with the number of files and the separated data block numbers in one file.

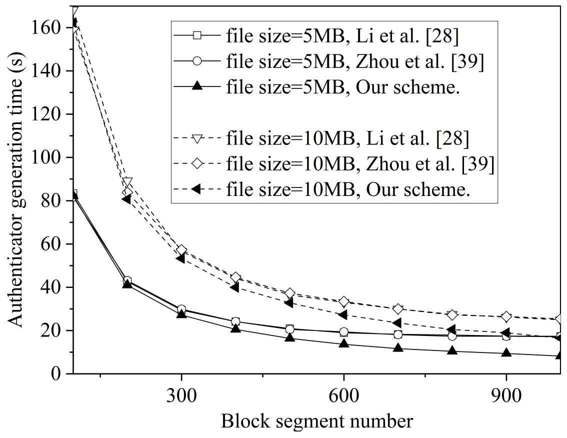


Fig. 4: The authenticator generation time with the block segment number and different file size.

the number of files containing the same keyword in experiment was set to 1. According to the results, we observed that the index generation time increased as the number of separated data block numbers grew. Our scheme demonstrated greater efficiency compared to [17], [19]. For example, when the separated data block numbers were configured to 500, the time required for index generation in schemes [17], [19] and our proposed approach was 23.134, 22.969, and 13.683 seconds, respectively. Compared to scheme [17], [19], the time cost was reduced by approximately 40% in our proposed approach. This is because we have reduced the hash-to-group mapping operations in the indexing process. The design ensured that the security remained uncompromised but improved the efficiency.

**—Evaluation time cost for proof generation and verification**.

During the experiment, two different configurations were tested: one with 300 challenge data blocks and the other with 460. In the case of 300 challenge data blocks, the ratio of corrupted data blocks was set to 1%, with a corresponding probability of detecting corrupted blocks at 95%. For the configuration with 460 challenge data blocks, the ratio of corrupted data blocks remained at 1%, but the probability of

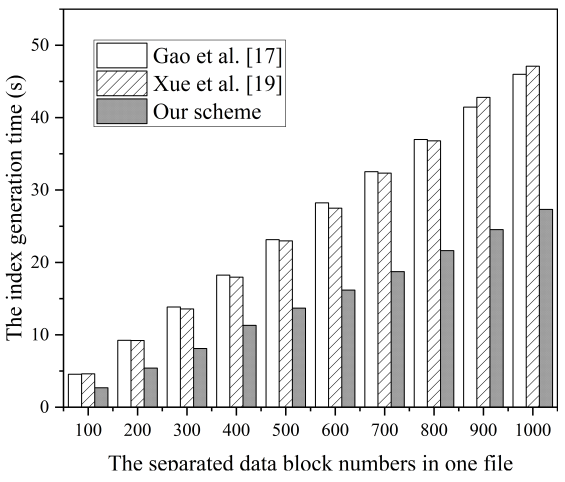


Fig. 5: The index generation time with the number of keyword and the separated data block numbers in one file.

detecting corrupted blocks increased to 99%. Since searchable auditing based on certificateless cryptography lacks comparable references in proof generation and verification, we solely conducted an estimation of the time cost evaluation based on our proposed approach.

Fig. 6 showed the computation costs when the challenge numbers was set to 300. In Fig. 6 (a), a comparison of time costs was presented for different numbers of sectors and different numbers of files containing the same keyword. The sector numbers in a block were varied from $1\times10^3$ to $1\times10^4$ with step size $1\times10^3$, and the number of files which contained the same keyword ranged from 20 to 100 with a step size of 20. It can be observed that the time costs were less affected by the number of sectors. When $\xi = 60$ and $s = 3000$, it required $160.662$ seconds, while it took $165.365$ seconds when $\xi = 60$ and $s = 7000$. That was significant for improving the efficiency on DO side. Additionally, the time costs grew linearly with the number of files which contained the same keyword. This increase was due to the rising number of aggregate files.

Fig. 6 (b) showed the computation costs when the challenge number was set to 300. In Fig. 6 (b), we conducted a comparison of time costs for various scenarios. This included different quantities of files containing the same keyword and varying numbers of sectors. We explored the impact of changing the number of files with the same keyword, ranging from 10 to 100 with increments of 10, and the number of sectors, which varied from $2\times10^3$ to $1\times10^4$ in increments of 2000. It demonstrated a linear increase in time costs as the number of files with the same keyword grew, while the time costs increased at a relatively slower rate with the expanding number of sectors. When $\xi = 50$ and $s = 4000$, it required $169.694$ seconds, while it took $301.564$ seconds when $\xi = 100$ and $s = 4000$.

Fig. 6 (c) illustrated the computation costs when the challenge numbers were set to 300. In Fig. 6 (c), a comparison of time costs was provided for different numbers of sectors and different numbers of files containing the same keyword. The number of sectors in a block was varied from $1\times10^3$ to $1\times10^4$ with a step size of $1\times10^3$, while the number of files containing the same keyword ranged from 20 to 100 with a step size of 20. It was showed that the time costs exhibited a linear relationship with the number of sectors in a block. That was because the CS needed to verify more information as the number of sectors increased. When $s = 2000$ and $\xi = 20$, it required $20.108$ seconds, while it took $70.345$ seconds when $s = 8000$ and $\xi = 20$.

Fig. 6 (d) showed the computation costs when the challenge numbers were set to 300. In Fig. 6 (d), the depicted results illustrate the time costs associated with various scenarios involving different numbers of files containing the same keyword and varying numbers of sectors. The number of files with the same keyword was tested across a range from 10 to 100 with increments of 10, and the number of sectors was assessed from $2\times10^3$ to $1\times10^4$ with increments of $2\times10^3$. It was noticeable that the time costs exhibited a non-linear relationship with the number of files which contained the same keyword. When $\xi = 60$ and $s = 3000$, it costed $28.383$ seconds while it costed $28.827$ seconds when $\xi = 80$ and $s = 3000$. This non-linear behavior was due to the verification time costs mainly involving some addition operations in a group concerning the number of files. The overhead was relatively lightweight.

In Fig.s 7 (a)-(d), we presented the computation costs associated with a challenge number set to 460. Fig. 7 (a) illustrated the time costs for various combinations of the number of sectors and the number of files containing the same keyword. The number of sectors in a block was adjusted from $1\times10^3$ to $1\times10^4$ with increments of $1\times10^3$, and the number of files with the same keyword ranged from 20 to 100 with increments of 20. The observations showed that the time costs were less affected by the number of sectors. When $\xi = 60$ and $s = 3000$, it costed $254.641$ seconds while it costed $259.987$ seconds when $\xi = 60$ and $s = 7000$. Additionally, the time costs exhibited a linear relationship with the number of files containing the same file. When $\xi = 20$ and $s = 5000$, it costed $82.67$ seconds, while it costed $264.833$ seconds when $\xi = 60$ and $s = 5000$.

In Fig. 7 (b), the results illustrated the time cost comparison for various numbers of files containing the same keyword and different numbers of sectors. The number of files which contained the same keyword ranged from 10 to 100 with increments of 10, and the number of sectors was adjusted from $2\times10^3$ to $1\times10^4$ with increments of $2\times10^3$. The results indicated that time costs increased linearly with the number of files which contained the same keyword and exhibited gentle growth with changes in the number of sectors. When $\xi = 30$ and $s = 2000$, it costed $84.028$ seconds while it costed $110.396$ seconds when $\xi = 30$ and $s = 6000$. Next, Fig. 7 (c) depicted the time cost comparison for various sectors and files containing the same keyword. The number of sectors in a block ranged from $1\times10^3$ to $1\times10^4$ with increments of $1\times10^3$, and the number of files which contained the same keyword varied from 20 to 100 with increments of 20. When $s = 5000$ and $\xi = 60$, it costed $48.584$ seconds while it costed $69.604$ seconds when $s = 8000$ and $\xi = 60$. The results demonstrated that time costs displayed a linear relationship with the number

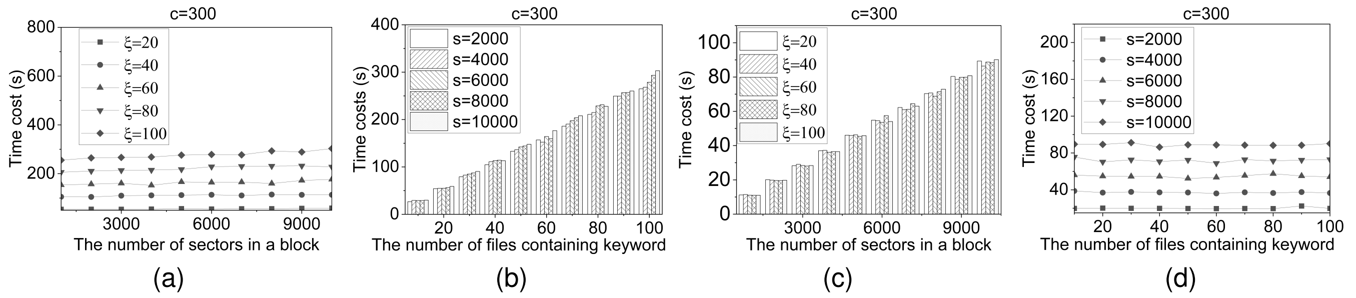


Fig. 6: The time cost when c=300 (a): with different numbers of sectors in a block in proof generation; (b): with different numbers of files containing the keyword in proof generation; (c): with different sectors in a block in proof verification; (d): with different numbers of files containing the keyword in proof verification.

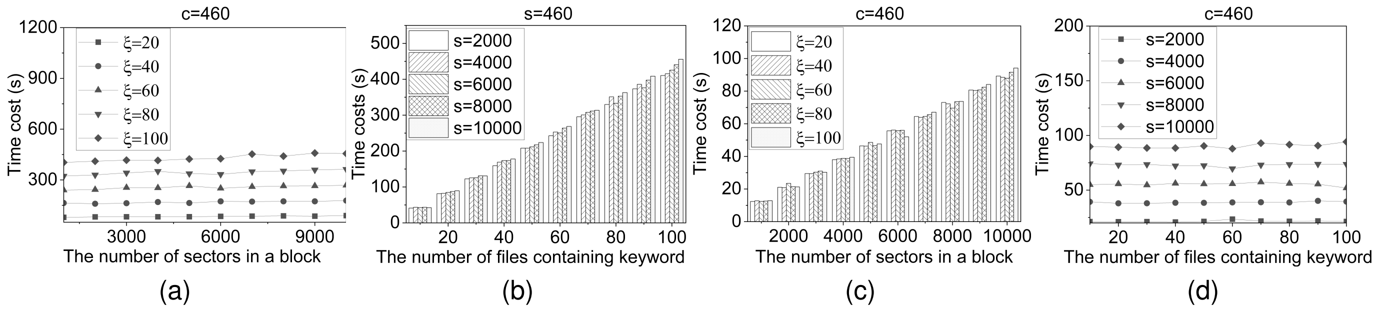


Fig. 7: The time cost when c=460 (a): with different numbers of sectors in a block in proof generation; (b): with different numbers of files containing the keyword in proof generation; (c): with different sectors in a block in proof verification; (d): with different numbers of files containing the keyword in proof verification.

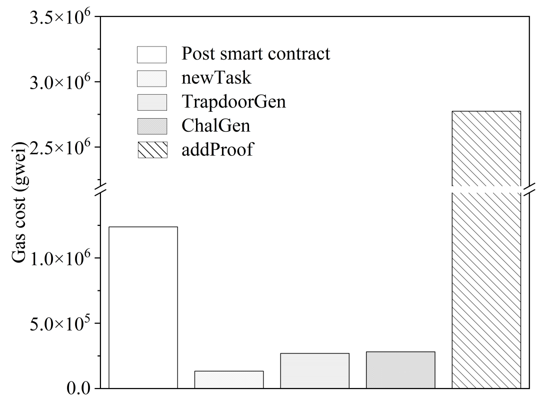


Fig. 8: Gas cost.

of sectors in a block. Fig. 7 (d) illustrated the time cost comparison for different numbers of files containing the same keyword and different numbers of sectors. The number of files which contained the same keyword ranged from 10 to 100 with increments of 10, and the number of sectors varied from $2\times10^3$ to $1\times10^4$ with increments of $2\times10^3$. The observations revealed that time costs exhibited a nonlinear relationship with the number of files which contained the same keyword. When $\xi = 60$ and $s = 3000$, it costed 30.33 seconds, while it costed 30.976 seconds when $\xi = 80$ and $s = 3000$.

**—Evaluation gas cost for smart contract**.

Fig. 8 showed the gas cost of the smart contract. We published the smart contract on a test chain from Ethereum. The test data were collected on December 20, 2023. In general, the more data sent to the blockchain, the more gas consumption were required. From the Figure, we observed that it costed 1237740 $gwei$ to post the smart contract. When comparing the gas consumption of the four functions, the function $newTask$ required the least gas. Next, the gas consumption in functions $TrapdoorGen$ and $ChalGen$ were similar. The function $addProof$ had the highest gas consumption. That was because this function needed to record more information, including the proof information and an index vector.

## VIII. Discussion and Future Work

**Inriching the functions.** The current scheme supports integrity auditing based solely on a single data owner. To enhance its functionality, we aim to explore how to extend the scheme to data sharing scenario. While numerous solutions exist in data sharing scenarios, existing approaches remain inadequate for large-scale similarity data sharing enviornment, necessitating the redefinition of security models and the exploration of privacy preservation challenges.

**Improving computation efficiency.** The verification cost is costly with large volumes of files since we adopt pairing-based cryptosystem. Reducing computation overheads is a significant challenge in designing efficient auditing schemes and our next rearch goal.

## IX. Conclusion

In this paper, we proposed a blockchain-assisted searchable integrity auditing for large-scale similarity data scheme. The proposed scheme achieved searchable auditing for uncertain numbers of similar data, improving the application wide of the auditing model. The proposed scheme did not rely on a centralized TPA but utilized blockchain combining with smart contract to enhance the credibility and transparency of the auditing process. The auditing process was facilitated through collaboration between CS and smart contracts, thus eliminating the requirement for TPA to remain continuously online awaiting responses from CS. Furthermore, the certificateless authenticator combined with the index matrix and structure were designed to achieve searchable auditing and reduce the overhead of key and certificate management. An arbitration resolution was introduced to deal with disputes between DO and CS. Theoretical and security analysis showed that the proposed scheme was efficient and secure.

## References

[1] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009–2030, 2020.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*. Alexandria Virginia, USA: Association for Computing Machinery, New York, United States, 2007, pp. 598–609.

[3] A. Li, Y. Chen, Z. Yan, X. Zhou, and S. Shimizu, "A survey on integrity auditing for data storage in the cloud: from single copy to multiple replicas," *IEEE Transactions on Big Data*, vol. 8, no. 5, pp. 1428–1442, 2020.

[4] H. Han, , S. Fei, Z. Yan, and X. Zhou, "A survey on blockchain-based integrity auditing for cloud data," *Digital Communications and Networks*, vol. 8, no. 5, pp. 591–603, 2022.

[5] W. Guo, S. Qin, F. Gao, H. Zhang, W. Li, Z. Jin, and Q. Wen, "Dynamic proof of data possession and replication with tree sharing and batch verification in the cloud," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 1813–1824, 2020.

[6] Y. Su, Y. Li, B. Yang, and Y. Ding, "Decentralized self-auditing scheme with errors localization for multi-cloud storage," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2838–2850, 2021.

[7] Z. Liu, L. Ren, R. Li, Q. Liu, and Y. Zhao, "Id-based sanitizable signature data integrity auditing scheme with privacy-preserving," *Computers & Security*, vol. 121, p. 102858, 2022.

[8] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Science China Information Sciences*, vol. 62, no. 3, pp. 1–16, 2019.

[9] Y. Yang, Y. Chen, F. Chen, and J. Chen, "Identity-based cloud storage auditing for data sharing with access control of sensitive information," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10 434–10 445, 2021.

[10] Y. Yang, Y. Chen, and F. Chen, "An efficient identity-based provable data possession protocol with compressed cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1359–1371, 2022.

[11] D. Liu, Z. Li, and D. Jia, "Secure distributed data integrity auditing with high efficiency in 5g-enabled software-defined edge computing," *Cyber Security and Applications*, vol. 1, p. 100004, 2023.

[12] L. Zhou, A. Fu, G. Yang, Y. Gao, S. Yu, and R. Deng, "Fair cloud auditing based on blockchain for resource-constrained iot devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4325–4342, 2022.

[13] C. Wang, X. Liu, H. Li, X. Di, L. Cong, S. Zhang, and H. Qi, "Smart contract-based integrity audit method for iot," *Information Sciences*, vol. 647, p. 119413, 2023.

[14] 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 Transactions on Network and Service Management*, vol. 19, no. 4, pp. 5333–5347, 2022.

[15] X. Li, S. Shang, S. Liu, K. Gu, M. Jan, X. Zhang, and F. Khan, "An identity-based data integrity auditing scheme for cloud-based maritime transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2556–2567, 2022.

[16] A. Hu, R. Jiang, and B. Bhargava, "Identity-preserving public integrity checking with dynamic groups for cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 4, pp. 1097–1110, 2018.

[17] X. Gao, J. Yu, Y. Chang, H. Wang, and J. Fan, "Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 3774–3789, 2021.

[18] M. Tian, Y. Zhang, Y. Zhu, L. Wang, and Y. Xiang, "Divrs: Data integrity verification based on ring signature in cloud storage," *Computers & Security*, vol. 124, p. 103002, 2023.

[19] J. Xue, S. Luo, Q. Deng, L. Shi, X. Zhang, and H. Wang, "Ka: Keyword-based auditing with frequency hiding and retrieval reliability for smart government," *Journal of Systems Architecture*, vol. 138, p. 102856, 2023.

[20] S. Li, C. Xu, Y. Zhang, Y. Du, and K. Chen, "Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 134–146, 2022.

[21] Y. Lin, J. Li, S. Kimura, Y. Yang, Y. Ji, and Y. Cao, "Consortium blockchain-based public integrity verification in cloud storage for iot," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3978–3987, 2021.

[22] Q. Zhao, S. Chen, Z. Liu, T. Baker, and Y. Zhang, "Blockchain-based privacy-preserving remote data integrity checking scheme for iot information systems," *Information Processing & Management*, vol. 57, no. 6, p. 102355, 2020.

[23] Y. Miao, Q. Huang, M. Xiao, and W. Susilo, "Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3663–3676, 2022.

[24] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "Sclpv: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 159–170, 2015.

[25] S. Li, C. Xu, Y. Zhang, Y. Du, A. Yang, X. Wen, and K. Chen, "Backdoor-resistant public data integrity verification scheme based on smart contracts," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 269–14 284, 2023.

[26] A. Juels and S. B. J. Kaliski, "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*. New York, NY, United States: Association for Computing Machinery, New York, United States, 2007, pp. 584–597.

[27] C. Yang, Y. Liu, F. Zhao, and S. Zhang, "Provable data deletion from efficient data integrity auditing and insertion in cloud storage," *Computer Standards & Interfaces*, vol. 82, p. 103629, 2022.

[28] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 356–365, 2019.

[29] Z. Li, Y. Li, B. Yang, and Y. Ding, "Algebraic signature-based public data integrity batch verification for cloud-iot," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 3184–3196, 2023.

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

[31] Y. Fan, H. Wu, and H. Paik, "Dr-bft: A consensus algorithm for blockchain-based multi-layer data integrity framework in dynamic edge computing system," *Future Generation Computer Systems*, vol. 124, pp. 33–48, 2021.

[32] W. Liang, Y. Liu, C. Yang, S. Xie, K. Li, and W. Susilo, "On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: A comprehensive survey," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–35, 2024.

[33] H. Guo, Y. Chen, X. Chen, Y. Huang, and Z. Zheng, "Smart contract code repair recommendation based on reinforcement learning and multi-metric optimization," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 4, pp. 1–31, 2024.

[34] S. Ismail, M. Nouman, H. Reza, F. Vasefi, and H. Zadeh, "A blockchain-based fish supply chain framework for maintaining fish quality and authenticity," *IEEE Transactions on Services Computing*, vol. 17, no. 5, pp. 1877–1886, 2024.

[35] G. Falazi, U. Breitenbücher, F. Leymann, and S. Schulte, “Cross-chain smart contract invocations: a systematic multi-vocal literature review,” *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–38, 2024.

[36] C. Sendner, L. Petzi, J. Stang, and A. Dmitrienko, “Large-scale study of vulnerability scanners for ethereum smart contracts,” in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2273–2290.

[37] D. Chen, Z. Liao, R. Chen, H. Wang, C. Yu, K. Zhang, N. Zhang, and X. Shen, “Privacy-preserving anomaly detection of encrypted smart contract for blockchain-based data trading,” *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 5, pp. 4510–4525, 2024.

[38] Z. Lin, J. Chen, J. Wu, W. Zhang, Y. Wang, and Z. Zheng, “Crpwarner: Warning the risk of contract-related rug pull in defi smart contracts,” *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1534–1547, 2024.

[39] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, “Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1118–1132, 2020.

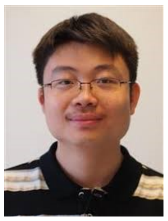
[40] Y. Tian, H. Tan, J. Shen, V. Pandi, B. Gupta, and V. Arya, “Efficient identity-based multi-copy data sharing auditing scheme with decentralized trust management,” *Information Sciences*, vol. 644, p. 119255, 2023.

**Ying Miao** received her M.S. degree from South China Agricultural University. She is now a PhD Student at School of Computer Science & Technology, Beijing Institute of Technology. She has published more than ten papers about blockchain, data security and machine learning. Her research interests include information security, blockchain and cloud computing.

**Keke Gai** received the Ph.D. degree in computer science from Pace University, New York, NY, USA. He is currently a Professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include cyber security, blockchain, AI security, and privacy computation. He has published more than 200 peer-reviewed papers in recent years and has been granted more than 10 Best Paper Awards. He is serving as an Editor-in-Chief of the journal Blockchains and an Editorial Board member of a few journals, including TDSC, JPDC, FGCS, etc. He is currently serving a few academic organizations, e.g., 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.

**Yu-an Tan** received the B.Eng. degree in Computer Software in 1991, Ph.D. in Computer Science in 2003. He has got teaching and research experience of more than 30 years, and has been a Professor in Beijing Institute of Technology since 2010. He is a senior member of the China Computer Federation. He contributes for peer reviewed 100+ journal papers and conference papers. He has received over 20 research funds from National Natural Science Foundation of China, National Key Research and Development Program of China, etc. His research areas include Artificial Intelligence Security, Cybersecurity and Storage Subsystem.

**Liehuang Zhu** received his Ph.D. degree in computer science from Beijing Institute of Technology, Beijing, China, in 2004, the M.E. (Master of Engineering) degree and B.E. (Bachelor of Engineering) degree from Wuhan University, Wuhan, China, in 2001 and 1998, respectively. He is currently a professor at School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China. He has published more than 4100 peer-reviewed journal or conference papers, including 10+ IEEE/ACM Transactions papers (IEEE TIFS, IEEE TII, IEEE TVT, IEEE TSG, Information Sciences, IEEE Network, Computer & Security, etc.). He has been granted a number of IEEE Best Paper Awards, including IWQoS 17’, TrustCom 18’. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing. He is a Senior Member of IEEE.



**Weizhi Meng** is a Full Professor in the School of Computing and Communications, Lancaster University, United Kingdom. He obtained his Ph.D. degree in Computer Science from the City University of Hong Kong. He was a recipient of the Hong Kong Institution of Engineers (HKIE) Outstanding Paper Award for Young Engineers/Researchers in both 2014 and 2017. He also received the IEEE ComSoc Best Young Researcher Award for Europe, Middle East, & Africa Region (EMEA) in 2020 and the IEEE ComSoc Communications & Information Security (CISTC) Early Career Award in 2023. His primary research interests are blockchain technology, cyber security and artificial intelligence in security including intrusion detection, blockchain applications, smartphone security, biometric authentication, and IoT security. He is senior member of IEEE.