Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.081399

ARTICLE

Blockchain-Based Transparent Certificateless Data Integrity Auditing with Enhanced Tag Security

Chao Zhang[1], Weidong Zhong[1], Xu An Wang[1], Weiwei Jiang[2,*], Ziteng Wang[2], Miao Tian[1], Jianhong Ling[1], Hangjiang Du[1] and Yunhui Duan[1]

[1]School of Cryptographic Engineering, Engineering University of People's Armed Police, Xi'an, China
[2]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding Author: Weiwei Jiang. Email: jww@bupt.edu.cn

Received: 03 March 2026; Accepted: 14 May 2026; Published: 15 June 2026

ABSTRACT: The integrity risks posed by data outsourcing in cloud storage have driven the development of remote data integrity auditing (RDIA) technologies. However, traditional schemes rely on trusted third-party auditors (TPAs), leading to potential collusion and single-point failure vulnerabilities. The integration of blockchain alleviates these issues through decentralization and transparency, yet existing blockchain-based certificateless auditing schemes still suffer from security flaws in the tag generation phase. Addressing the tag forgery vulnerability in Miao et al.'s scheme, which stems from the absence of random parameters in the hash function input, this paper proposes a lightweight enhancement mechanism: incorporating a random factor into the hash input during tag generation to ensure dynamic unforgeability of tags. While retaining the efficiency advantages of the original framework, the improved scheme achieves resistance against tag forgery, proof forgery, and collusion attacks under the Computational Diffie-Hellman (CDH) and Discrete Logarithm (DL) hardness assumptions, validated through rigorous formal proofs. Experimental performance analysis demonstrates that the proposed enhanced scheme introduces negligible computational overhead, providing a secure, practical, and transparent auditing solution for multi-cloud storage environments.

KEYWORDS: Blockchain; tag security; certificateless cryptography; data integrity auditing; cloud storage

1 Introduction

In the era of vigorous development of the digital economy, data has become a core strategic asset driving technological innovation, decision-making and value creation. As a critical infrastructure supporting the explosive growth of massive data, cloud storage is deeply integrated into the business processes of modern enterprises and the digital life of individual users by virtue of its outstanding scalability, low-cost advantages and cross-platform convenient access. Cloud storage not only greatly alleviates the pressure on local storage resources, but also realizes real-time global data collaboration and sharing, delivering unprecedented convenience. However, this outsourced storage mode with separated data ownership and management rights significantly reduces users' direct physical control over cloud data while creating value, and triggers a series of severe security risks. Among these challenges, ensuring data integrity stands out as the most essential and critical issue. After data is uploaded to the cloud, users have to rely entirely on the cloud service provider (CSP) to guarantee data accuracy and consistency. Nevertheless, cloud data is highly vulnerable to loss, corruption and malicious tampering due to non-human factors such as hardware failures and system vulnerabilities, as well as human threats including internal operational negligence

and external malicious attacks. For highly sensitive data such as financial transaction records, medical archives and intellectual property documents, any minor damage to data integrity may lead to catastrophic consequences. Therefore, achieving efficient and reliable integrity verification for outsourced data without compromising the convenience of cloud storage has become a research hotspot attracting widespread attention from both academia and industry. Remote Data Integrity Auditing (RDIA) technology has emerged to address this demand. Its core principle is as follows: resource-constrained Data Owners (DOs) can entrust a Trusted Third-Party Auditor (TPA) to initiate periodic or on-demand auditing challenges to the Cloud Server (CS) through lightweight cryptographic protocols. The TPA verifies the proof returned by the cloud server and determines the integrity and credibility of outsourced data without downloading the complete raw data. Since Ateniese et al. first proposed the pioneering Provable Data Possession (PDP) model, numerous optimized schemes supporting dynamic data updates, privacy preservation and multi-cloud collaboration have been proposed in this field. Although the functional capabilities of remote data integrity auditing frameworks have become increasingly mature, the commonly adopted centralized third-party auditing architecture suffers from fundamental security bottlenecks. On the one hand, third-party auditing nodes are prone to single points of failure, which directly affects the overall availability of auditing services. On the other hand, the inescapable risk of collusion attacks between auditors and cloud servers persists. Once the two collude, the auditor may leak challenge information in advance and forge verification results, rendering the entire auditing mechanism invalid and seriously undermining user trust in data integrity. To completely eliminate reliance on trusted third-party auditors, researchers have introduced blockchain technology to reconstruct remote data integrity auditing architectures. Under the assumption that most nodes are honest, blockchain features decentralization, immutability and traceability, providing a novel and feasible solution for constructing a truly trusted auditing system. Based on this architecture, smart contracts deployed on the blockchain can automatically complete core operations throughout the entire process, including audit challenge generation, proof verification and on-chain result storage. This reduces dependence on centralized intermediaries to a certain extent and builds a more reliable auditing execution environment. Meanwhile, to simplify key management, avoid the high lifecycle overhead of complex certificates in the traditional Public Key Infrastructure (PKI), and resolve the inherent key escrow vulnerability of Identity-Based Cryptography (IBC), the Certificateless Cryptography (CLC) scheme has emerged as a competitive alternative. It strikes a favorable balance between key management complexity and security, requiring no digital certificates and eliminating the security hazard that the Key Generation Center (KGC) fully controls users' private keys. Against this research background, Miao et al. proposed a blockchain-based transparent certificateless data integrity auditing scheme in recent years. This scheme replaces traditional centralized third-party auditors with smart contracts to achieve openness, fairness and automation of auditing procedures, effectively overcoming the inherent defects of traditional public key and identity-based cryptosystems, and holding promising application prospects in multi-cloud storage scenarios. Nevertheless, in-depth analysis reveals potential security flaws in the core data tag generation module of this scheme. In its tag construction algorithm, the hash input for binding data block metadata only contains static or semi-static public parameters such as file names, block indices and public keys, while lacking critical dynamic random factors. Under the Chosen Message Attack (CMA) model, a malicious cloud server can generate forged tags through offline precomputation, bypass the integrity verification mechanism, and conduct covert data tampering attacks. Targeting the above security vulnerabilities, this paper proposes a lightweight improved scheme with enhanced tag security. The core design is concise and efficient: a dynamic random parameter $\boldsymbol{T}$ is introduced into the hash operation during tag generation to ensure the uniqueness and unpredictability of each data tag, fundamentally improving the resistance to relevant attacks. The main contributions of this paper are summarized as follows: 1. This paper accurately identifies and formally analyzes the security flaws of existing schemes. It systematically dissects the tag forgery vulnerabilities in Miao's

scheme, strictly defines three types of adversaries with different capabilities and their attack targets under standard cryptographic adversary models, and clarifies the optimization direction for subsequent security enhancement design. 2. An efficient and universal security repair mechanism is designed. The improved scheme only adds a single hash operation to the original tag generation process, resulting in extremely low computational and communication overhead. Under established security assumptions, it can resist various attacks including tag forgery, proof forgery and multi-party collusion, possessing high engineering practical value. 3. Rigorous formal security proofs are provided. Based on the Computational Diffie-Hellman (CDH) and Discrete Logarithm (DL) hardness assumptions, strict reduction proofs are conducted to verify that the improved scheme satisfies core security properties such as Existential Unforgeability under Chosen Message Attacks (EUF-CMA) and auditing correctness. 4. Comprehensive performance evaluations are conducted. Multiple comparative experiments are carried out to quantitatively measure the operational overhead of the proposed scheme. Horizontal comparisons with baseline schemes verify the efficiency and feasibility of the proposed scheme in real multi-cloud storage environments. The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces preliminary knowledge and the system model. Section 4 reviews the original scheme in detail and identifies its security defects. Section 5 elaborates on the specific design of the improved scheme. Section 6 analyzes the correctness of the proposed scheme. Section 7 presents formal security proofs. Section 8 demonstrates experimental simulations and performance evaluation results. Section 9 concludes the full text and prospects future research directions.

2 Related Work

The development of remote data integrity auditing technology has witnessed a complete evolution from centralization to decentralization, static verification to dynamic updating, and single-function design to multi-objective collaboration. Combined with representative domestic and international research achievements, this chapter systematically reviews the research progress of two mainstream directions: traditional centralized auditing and blockchain-based decentralized auditing.

2.1 Traditional Public Auditing Schemes

The rapid popularization of cloud storage relies on standardized definitions and systematic architectural research. Mell and Grance [1] released the authoritative cloud computing definition formulated by the National Institute of Standards and Technology (NIST), clarifying the basic service modes and core characteristics of cloud computing. Armbrust et al. [2] comprehensively summarized the overall architecture, technical advantages and potential risks of cloud computing, laying a theoretical foundation for subsequent research on outsourced storage. With the large-scale migration of enterprise and user data to the cloud, security issues caused by the separation of data ownership and storage rights have become increasingly prominent. Hashizume et al. [3] systematically sorted out various security threats in cloud computing scenarios and pointed out that data integrity damage is a critical security risk that urgently needs to be addressed. The foundational work in the field of data integrity auditing was initiated by the Provable Data Possession (PDP) model proposed by Ateniese et al. [4]. Leveraging homomorphic authenticators, this scheme realizes lightweight remote data verification without downloading complete files, pioneering the research paradigm of public auditing. However, the original PDP scheme only supports static data verification and cannot repair corrupted data. To address this limitation, Juels and Kaliski [5] proposed the Proof of Retrievability (PoR) model. Combined with error-correcting coding technology, it supports the recovery of partially corrupted data while completing integrity auditing. On this basis, Shacham and Waters [6] constructed a streamlined and efficient PoR protocol under standard model assumptions, greatly reducing the communication overhead of auditing. In practical cloud service scenarios, users frequently

modify, insert and delete cloud data. To meet the demand for dynamic data, Wang et al. [7] designed a public auditing scheme supporting full-dimensional dynamic updates by combining Merkle Hash Trees, enabling iterative data updates and real-time integrity verification. Facing the needs of multi-user collaborative office in cloud sharing scenarios, Yuan and Yu [8] proposed a multi-user-oriented integrity detection protocol to optimize the collaborative editing and joint auditing process of shared data. For distributed multi-cloud storage architectures, Wang et al. [9] adopted a fragmented distributed data storage strategy and built a high-fault-tolerant cross-cloud auditing mechanism to improve the operational reliability of heterogeneous cloud storage systems. Ali et al. [10] summarized the technical framework, mainstream models and existing limitations of outsourced data auditing in the form of reviews, and comprehensively combed the development context of traditional auditing technologies. Privacy preservation and lightweight design are two core optimization directions for traditional auditing schemes. Wang et al. [11] introduced data blinding and random masking technologies to construct a privacy-preserving public auditing scheme, effectively preventing the leakage of users' sensitive data during auditing. Targeting resource-constrained scenarios such as mobile terminals and edge devices, Yoosuf and Anitha [12] proposed a lightweight dual auditing protocol LDuAP, which streamlines complex cryptographic operations to adapt to the computing power constraints of low-power devices. Zheng et al. [13] reviewed the development status of blockchain, providing theoretical support for decentralized auditing. Yue et al. [14] conducted research on data integrity verification in edge-cloud collaboration scenarios, further enriching the cross-domain storage security auditing system. Huang et al. [15] constructed a collaborative cloud data auditing architecture based on distributed collaboration ideas. From the perspective of auditing behavior constraint, Miao et al. [16] designed an incentive-based auditing protocol to restrain the negative behavior of auditing nodes. Liu et al. [17] optimized on-chain auditing processes by adopting blockchain expansion technology. Zhang et al. [18] focused on multi-replica storage scenarios and proposed a decentralized and efficient multi-replica auditing scheme. Zhu et al. [19] designed a universal dynamic auditing service framework for cloud outsourced data. Yang and Jia [20] further improved dynamic auditing protocols to enhance auditing security and efficiency in complex cloud environments. Zhou et al. [21] constructed a quantum-resistant cloud data PDP scheme by integrating lattice cryptography. Yang et al. [22] designed a lightweight provable data possession scheme specifically for mobile terminals. Li et al. [23] realized privacy-preserving remote data integrity detection by integrating identity-based cryptography. Yuan et al. [24] combined identity-based cryptography with blockchain to achieve cross-environment trusted data verification. Li et al. [25] built a blockchain-assisted batch public auditing mechanism for big data scenarios. Xu et al. [26] realized privacy-friendly data auditing in consortium blockchain scenarios based on zero-knowledge proofs. Chang et al. [27] optimized the efficiency of multi-replica verification in multi-cloud architectures to improve cross-cloud storage auditing performance. Li and Hu [28] designed a smart contract-driven multi-party collaborative auditing mechanism for multiple audit participants. Tahir et al. [29] proposed a lightweight blockchain-based security authentication and data auditing framework for the Internet of Medical Things. Nevertheless, most public auditing schemes rely on a trusted third-party auditor (TPA), and inherent security risks such as single point of failure of auditing nodes and collusion between cloud servers and auditors cannot be fundamentally eliminated.

2.2 Blockchain-Based Public Auditing Schemes

To solve the trust dilemma of centralized third-party auditing, blockchain technology with decentralization, immutability and traceability has been gradually applied in the field of cloud data auditing. Miao et al. [30] constructed a transparent certificateless cloud storage auditing scheme, which completely replaces centralized third-party auditors with smart contracts to realize full-process auditing automation. Nweje [31] illustrated the application value of blockchain in security auditing, traceability and tamper resistance, and

pointed out the development trend of decentralized auditing technology. Based on blockchain auditing, Li et al. [32] integrated encrypted deduplication technology to reduce cloud storage costs while ensuring data integrity. Zhu et al. [33] proposed a lightweight certificateless auditing scheme without third-party auditors, which further reduces system overhead and realizes lightweight decentralized auditing. From the perspective of data compliance supervision, Wang et al. [34] built a GDPR-compliant blockchain auditing framework to meet the standardized management requirements of data security. Shen et al. [35] combined keyword search with remote auditing to realize rapid positioning and dynamic verification of cloud files. Tu et al. [36] optimized traditional multi-replica auditing algorithms to improve the efficiency and stability of redundant data verification in multi-cloud environments. Kumar and Bandanadam [37] introduced an improved ElGamal encryption algorithm to strengthen the encryption protection capability of blockchain auditing systems and enhance the attack resistance in decentralized storage scenarios. In recent years, research has been further advanced to promote the engineering implementation of auditing systems and adapt to complex application environments. Wang et al. [38] proposed the SStore provable data auditing platform, which significantly optimizes the overall system operational efficiency while guaranteeing security, and advances the development of auditing mechanisms toward platformization and practical application. Liu et al. [39] introduced a file access prediction mechanism in the shared data auditing scenario, combining data behavior analysis with integrity verification to improve audit response efficiency. With the continuous evolution of blockchain auditing technology, research priorities have gradually expanded to privacy protection, deduplication mechanisms and functional expansion. Zhang et al. [40] integrated blockchain with privacy-preserving deduplication technology, which reduces redundant storage overhead while realizing data integrity verification. Targeting the multi-replica dynamic data environment, Zhou et al. [41] designed a certificate-based multi-replica auditing scheme supporting data updates, which enhances system reliability and improves dynamic adaptability. Miao et al. [42] further expanded the auditing functions and proposed a blockchain-assisted provable data possession scheme supporting multi-keyword search, realizing the integration of data retrieval and integrity verification. Vijayakumar et al. [43] introduced a fair payment mechanism into the blockchain auditing system, combining data deduplication with economic incentives to improve the sustainable operation capability of the system. For special application scenarios, Xu et al. [44] proposed an auditing mechanism balancing privacy protection and transparent deduplication for UAV cloud storage environments, providing new ideas for data security in resource-constrained scenarios. Some existing certificateless blockchain auditing schemes attempt to reduce or replace the centralized third-party auditor (TPA), and achieve satisfactory performance in operational efficiency, privacy protection and scalability. However, they generally ignore the anti-forgery design of data verification tags. The tag generation process lacks dynamic random factors, enabling malicious cloud servers to precompute offline and forge legitimate tags. Focusing on this core security defect, this paper optimizes the tag generation algorithm on the basis of existing mainstream certificateless blockchain auditing schemes, and proposes an improved auditing scheme with lightweight features and high security.

3 Preliminary Knowledge and Background

3.1 Bilinear Pairing

Let $\mathbb{G}$ be an additive cyclic group and $\mathbb{G}_T$ be a multiplicative cyclic group, both of prime order q, where the Discrete Logarithm Problem (DLP) in $\mathbb{G}$ is intractable. A bilinear pairing is a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying:

- **Bilinearity:** For any $P, Q \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q^*$, $e(aP, bQ) = e(P, Q)^{ab}$.
- **Non-degeneracy:** There exist $P, Q \in \mathbb{G}$ such that $e(P, Q) \neq 1_{\mathbb{G}_T}$ ($1_{\mathbb{G}_T}$ is the identity element of $\mathbb{G}_T$).
- **Computability:** An efficient algorithm exists to compute $e(P, Q)$ for any $P, Q \in \mathbb{G}$.

Throughout this paper, $z \in \mathbb{Z}_q^*$ denotes the master secret key held by the Key Generation Center (KGC), and the corresponding system public key is defined as

$$P_{\text{pub}} = zP, \tag{1}$$

where P is the generator of the additive cyclic group $\mathbb{G}$. This definition is used consistently in the tag generation, proof generation, and proof verification algorithms.

3.2 Hard Computational Problems

3.2.1 Computational Diffie-Hellman (CDH) Problem

Given P, aP, bP ($a, b \in \mathbb{Z}_q^*$ unknown), it is computationally infeasible for a Probabilistic Polynomial-Time (PPT) adversary to compute abP.

3.2.2 Discrete Logarithm (DL) Assumption

Given a generator $P \in \mathbb{G}$ and $Q = aP$ (where $a \in \mathbb{Z}_q^*$ is unknown), a PPT adversary cannot compute a with non-negligible probability.

3.3 System Model

The role definitions of each entity are shown in Fig. 1: The core entities of the scheme and their core functional responsibilities are clearly defined as follows, with a clear distinction between the functional boundaries of the Third-Party Auditor (TPA) and blockchain smart contracts:

KGC Key Generation Center: A semi-trusted entity responsible for generating system-level public parameters and partial private keys for Data Owners (DOs) and strictly abides by the certificateless cryptography design principle—no full control over the user's complete private key—to avoid the key escrow vulnerability of Identity-based Cryptography (IBC).

DO Data Owner: The legitimate owner of the data with limited local storage and computing resources. The DO divides the data into blocks, generates enhanced tags using the modified tag generation algorithm, and uploads the data/tags to the CS. It delegates auditing tasks to the TPA and monitors the audit logs on the blockchain.

TPA A professional computing entity with strong cryptographic computing capabilities, serving as an auxiliary verification entity for the scheme. Its core functions include performing high-efficiency professional proof verification calculations based on the challenge information and proof data on the blockchain; conducting off-chain recheck of the audit results generated by the smart contract to ensure verification accuracy; and sending early warnings for abnormal audit results (such as data tampering) to the DO and other relevant entities.

CS Cloud Server: A storage service provider with abundant resources responsible for storing the DO's data and tags. When challenged by the TPA, the CS generates integrity proofs using the stored data and tags.

Blockchain The core decentralized execution entity of the auditing scheme, responsible for the core on-chain auditing logic and data storage. Its core functions include generating decentralized and tamper-proof audit challenge information based on the DO's secret value and timestamp; receiving the integrity proof submitted by the CS; providing the verification equation and basic computing support for proof verification; recording all audit processes (challenge generation, proof submission) and final results on the blockchain; and ensuring the immutability and public traceability of audit logs through the blockchain's consensus mechanism.

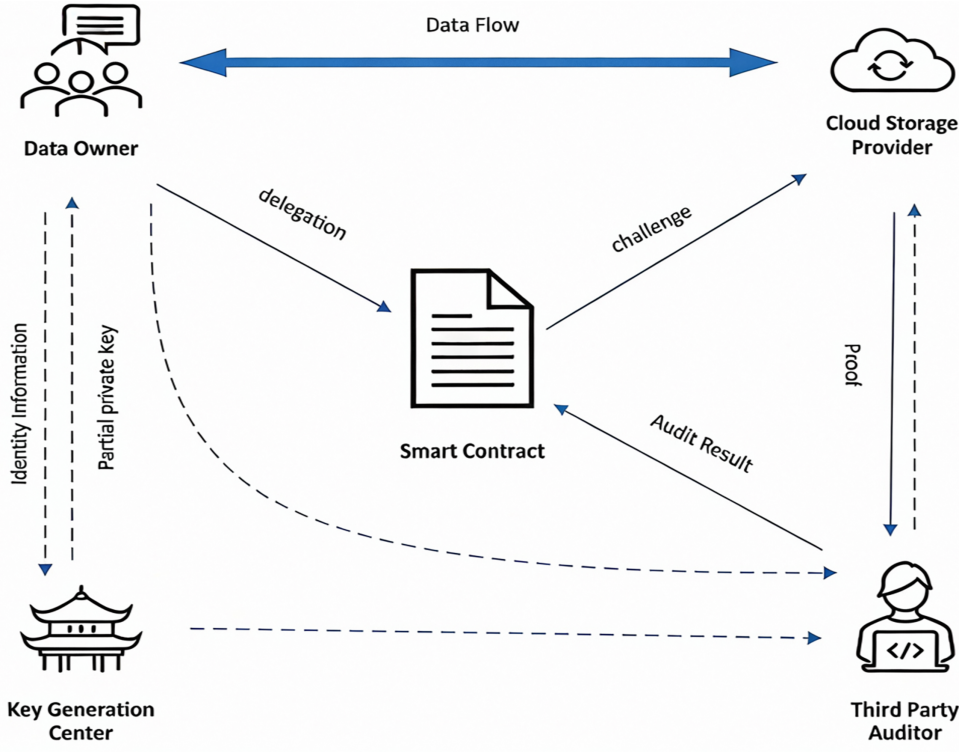

Figure 1: System model diagram.

Decentralization of the Auditing Process

The scheme proposed in this paper realizes the complete decentralization of the cloud storage data integrity auditing process, breaking the dependence on the centralized TPA in traditional remote data integrity auditing schemes. In this scheme, the TPA is only an optional auxiliary computing entity that provides professional cryptographic computing capability support for the auditing process, and its existence does not affect the essential decentralized characteristics of the scheme.

The blockchain smart contract is designed with a complete and independent audit verification logic, which integrates all core functions required for the auditing process: it can independently generate decentralized challenge information without relying on any centralized entity, can execute the proof verification equation according to the pre-deployed code logic, and can store the audit results and logs on the blockchain in an immutable manner. Even if the TPA is completely removed from the system, each blockchain node can complete the full process of data integrity auditing (challenge generation→proof reception→proof verification→result recording) through the deployed smart contract, and the audit results generated by the smart contract still have the properties of correctness, immutability and public verifiability.

The TPA's participation in the auditing process is only to improve the efficiency of audit verification: for large-scale multi-cloud storage auditing scenarios, the cryptographic computing of proof verification will bring a certain load to the blockchain nodes; the TPA with strong professional computing capabilities can undertake the heavy proof verification calculation work, and feed back the verification results to the blockchain for on-chain recording, which effectively reduces the computing load of blockchain nodes and improves the overall throughput of the auditing system. Whether the TPA is involved or not, the core decentralized audit logic of the scheme remains unchanged, which fully guarantees the decentralization and trustworthiness of the auditing process.

3.4 Threat Model

3.4.1 Fundamentals and Assumptions of Threat Modeling

This paper adopts the standard cryptographic adversary model, considering Probabilistic Polynomial-Time (PPT) adversaries. We conduct security analysis based on the following key assumptions:

- **Blockchain Security Assumption:** The blockchain network itself is secure, and its consensus mechanism can ensure the immutability and final consistency of the ledger. Smart contract code is assumed to be correctly implemented without vulnerabilities.
- **KGC Trust Assumption:** The Key Generation Center (KGC) is modeled as an "honest-but-curious" entity, meaning it will honestly execute the protocol but may attempt to infer user data or private keys using the partial private key information it holds.
- **Communication Security Assumption:** All communication channels (including KGC-DO, DO-CS, DO-TPA, etc.) are assumed to be secure. Adversaries can only obtain information by eavesdropping or tampering with channel contents, but cannot directly access secret keys.
- **Computational Assumption:** Based on the Computational Diffie-Hellman (CDH) and Discrete Logarithm (DL) hardness assumptions, it is believed that these mathematical problems cannot be solved within polynomial time.

3.4.2 Formalized Description of Adversary Capabilities and Attack Goals

To enhance the rigor of security analysis, we formally define three types of PPT adversaries with distinct capabilities and clear attack goals based on the certificateless cryptography adversary model and the characteristics of blockchain-based auditing systems:

- **Type I Adversary $\mathcal{A}_1$:**
  Capabilities (Formalized):
  1. Cannot access the KGC's master key z (i.e., $A_1 \notin \mathcal{K}$, where $\mathcal{K}$ denotes the set of entities with master key access).
  2. Has the authority to replace the Data Owner's (DO's) public key PK_{ID_u} with any arbitrary value PK'_{ID_u} (i.e., $\forall ID_u$, A_1 can output PK'_{ID_u} to replace the original PK_{ID_u}).
  3. Can fully control one or more malicious Cloud Servers (CSs), enabling it to tamper with stored data blocks m_i, forge tags σ'_i, and manipulate the proof generation process.
  4. Can eavesdrop on all public channel communications, including system public parameters pp, user public keys PK_{ID_u}, uploaded data tags σ_i, and on-chain audit logs (challenge seeds, proofs, results).
  5. Can perform polynomial-time queries, including:
     Partial key query: For any identity $ID_u \neq ID_u^*$ (target identity), query the KGC's partial private key D_{ID};
     Tag query: For any data block m_i and identity ID, query the corresponding tag σ_i generated by the DO.

  Attack Goal: Without obtaining the DO's complete private key $Sk_{ID_u} = (x_u, D_u)$ and original data blocks m_i, A_1 aims to forge a valid tag σ'_i for a tampered data block $m'_i \neq m_i$ such that the tag passes the TPA's verification (i.e., $\text{Verify}(pp, PK'_{ID_u}, m'_i, \sigma'_i, T') = \text{Accept}$), thereby bypassing the integrity auditing mechanism.
- **Type II Adversary $\mathcal{A}_2$:**
  Capabilities (Formalized):
  1. Can access the KGC's master key z (i.e., $A_2 \in \mathcal{K}$), enabling it to compute the partial private key $D_{\text{ID}} = zH_1(\text{ID}, P_{\text{pub}})$ for any user identity ID.

2. Cannot modify the DO's public key PK_{ID_u} (i.e., PK_{ID_u} is fixed once generated by the DO).
3. Can collude with the CS to tamper with stored data, forge tags, and manipulate audit proofs.
4. Can eavesdrop on all public and private channel communications (including the partial private key D_u transmitted from KGC to DO).
5. Can perform polynomial-time queries, including:
   Secret key query: For any identity $\mathrm{ID}_u \neq \mathrm{ID}_u^*$, query the complete private key $\mathrm{Sk}_{\mathrm{ID}} = (x_{\mathrm{ID}}, D_{\mathrm{ID}})$;
   Tag query: For any data block m_i and identity ID, query the corresponding tag σ_i.

Attack Goal: Without obtaining the DO's local private key x_u, A_2 aims to forge a valid tag σ_i' for a tampered data block m_i' or collude with the CS to generate a forged audit proof that passes verification, thereby achieving undetectable data tampering.

- **Type III Adversary** $\mathcal{A}_3$:
  Capabilities (Formalized):
  1. Represents a collusive alliance consisting of one or more malicious CSs and blockchain miners.
  2. Can tamper with stored data blocks m_i and tags σ_i on the CS, and forge integrity proofs $\mathrm{proof}' = (\delta', \mu', \overline{W}', T')$.
  3. Can manipulate blockchain challenge generation by colluding with miners:
     Biasing challenge block selection (e.g., avoiding tampered blocks by manipulating pseudorandom permutation π_{key});
     Delaying or modifying on-chain challenge seeds k_{f1}, k_{f2} (within the limits of blockchain consensus latency).
  4. Can access all system public parameters, user public information, and on-chain data (audit logs, transaction records).
  5. Cannot access the DO's private key $\mathrm{Sk}_{\mathrm{ID}_u}$ or KGC's master key z.

  Attack Goal: After tampering with data blocks (modification, insertion, deletion), A_3 aims to generate a forged audit proof proof' that passes the TPA's verification (i.e., $\mathrm{ProofVerify}(pp, \mathrm{proof}', k_{f1}, k_{f2}) = \mathrm{Accept}$), and manipulate on-chain challenge information to evade detection, thereby concealing data integrity breaches.

The improved scheme also needs to address the following additional threats:

- **Tag Forgery Attack:** Adversaries use precomputed static hash values $H_3(\mathrm{fname}||\mathrm{id}_i||Q_u)$ to forge valid tags for tampered data. Formally, given $h_i = H_3(\mathrm{fname}||\mathrm{id}_i||Q_u)$ (precomputed offline), the adversary constructs $\sigma_i' = m_i' D_u + x_u H_2(\mathrm{ID}_u||\mathrm{id}_i||p_{k_i}||T') + r' h_i$ and submits $T' = r'P$ to pass verification.
- **Collusion Attack:** Malicious CSs collude with miners to generate biased challenge information (e.g., avoiding tampered challenge blocks) and use forged tags to pass verification.
- **Privacy Leakage:** Adversaries infer sensitive data patterns (e.g., file structure, block importance) from static hash inputs $H_3(\mathrm{fname}||\mathrm{id}_i||Q_u)$. Formally, given a sequence of hash values $\{h_i\}$, the adversary infers the correlation between id_i and data sensitivity, leading to privacy breaches.

3.5 Design Goals

The design goals of the proposed scheme are as follows:

- **Audit Correctness:** If the valid proof generated by the CS passes TPA verification, the stored data must be intact (i.e., consistent with the data uploaded by the DO).
- **Privacy Protection:** During the auditing process, the TPA, CS, and other entities cannot access the DO's plaintext data or sensitive identity information.

- **Collusion Resistance:** The scheme resists collusion among any combination of entities (such as TPA-CS, CS-miners, KGC-CS) to prevent data tampering or manipulation of audit results.
- **Transparency:** All auditing processes (challenge generation, proof submission, verification) and results are recorded on the blockchain, ensuring public verifiability and tamper-proofing.
- **Tag Unforgeability:** Even if system parameters and public information are known, no PPT adversary can forge a valid tag for a data block without obtaining the DO's private key and real data.

4 Review of the Original Scheme

4.1 Process Overview

The scheme proposed by Miao et al. is based on certificateless cryptography and blockchain smart contracts, aiming to achieve decentralized, transparent, and TPA-free data integrity auditing. Its core idea is: using smart contracts to replace traditional TPAs for challenge generation; utilizing certificateless signatures to avoid PKI certificate management and IBC key escrow issues; and recording all audit logs on the chain to ensure traceability and immutability. The system auditing process includes:

Setup → PartialKeyGen → KeyGen → TagGen → Challenge (on-chain) → ProofGen → ProofVerify.

4.2 Tag Generation Algorithm (TagGen)

The most critical phase of this scheme is tag generation. For a file $M = \{m_1, m_2, \ldots, m_n\}$, the DO randomly selects $r \in \mathbb{Z}_q^*$ and computes $T = rP$. For each data block m_i, computes the tag:

$$\sigma_i = m_i D_u + x_u H_2(\mathrm{ID}_u \| \mathrm{id}_i \| \mathrm{pk}_i \| T) + r H_3(\mathrm{fname} \| \mathrm{id}_i \| Q_u) \tag{2}$$

where D_u is the KGC-generated partial private key, x_u is the DO's local private key, $Q_u = H_1(\mathrm{ID}_u)$, $\mathrm{pk}_u = x_u P$ and $T = rP$ is the random parameter. The DO uploads $\{m_i, \sigma_i\}_{i=1}^n$ and T to the CS.

4.3 Challenge and Verification Mechanism

Challenge generation is performed by smart contracts based on the secret value sv submitted by the DO, timestamp t, filename, etc., to generate pseudorandom seeds k_{f1} and k_{f2}, which further determine the challenge block indices and weights.

Proof generation involves the CS aggregating the tags and data of the challenged blocks to generate (δ, μ, W, T). Here, $P_{\mathrm{pub}} = zP$ denotes the system public key generated by the KGC during the Setup phase, where z is the KGC's master secret key and P is the generator of $\mathbb{G}$. The verification equation is:

$$e(\delta, P) = e(\mu Q_u - W, P_{\mathrm{pub}}) \cdot e\left(\sum_{i \in I} v_i H_2(\mathrm{ID}_u \| \mathrm{id}_i \| \mathrm{pk}_u \| T), \mathrm{pk}_u\right) \cdot e\left(\sum_{i \in I} v_i H_3(\mathrm{fname} \| \mathrm{id}_i \| Q_u), T\right). \tag{3}$$

5 Analysis of Attack

5.1 Tag Forgery Attack

Adversaries (such as malicious CSs or external adversaries) know public parameters, filenames fname, block id_i, and user public keys Q_u. Since the input of $H_3(\mathrm{fname} \parallel \mathrm{id}_i \parallel Q_u)$ is completely static, adversaries can precompute this hash value offline, denoted as: $h_i = H_3(\mathrm{fname} \parallel \mathrm{id}_i \parallel Q_u)$.

It should be emphasized that a malicious CS cannot legitimately compute a fresh tag in the same way as the DO executes the TagGen algorithm, because the CS does not possess the DO's complete private key, especially the local secret key x_u and the partial private key D_u. Therefore, the term "forged tag" in this

section does not refer to a normally generated cryptographic signature. Instead, it refers to an algebraically constructed or synthesized tag/proof component whose purpose is to satisfy the public verification equation by exploiting the linear structure of the original scheme.

The attack steps are as follows: The user uploads real data m_i and tags σ_i; the malicious CS tampers with the data to $m_i' \neq m_i$; the CS attempts to construct a new tag/proof component σ_i' to pass verification. Since h_i is known, the CS can select any $r' \in \mathbb{Z}_q^*$, set $T' = r'P$, and attempt to synthesize a forged tag/proof component with the following algebraic target form rather than compute a legitimate signature through TagGen:

$$\sigma_i' \overset{\text{alg}}{=} m_i' D_u + x_u H_2(\mathrm{ID}_u \parallel \mathrm{id}_i \parallel p_{k_i} \parallel T') + r' h_i. \tag{4}$$

The notation $\overset{\text{alg}}{=}$ indicates an algebraic relation required for satisfying the verification equation. It does not imply that the malicious CS can directly evaluate the secret-key-dependent terms $m_i' D_u$ and $x_u H_2(\cdot)$. In other words, σ_i' is not a valid tag generated by the DO, but an algebraically synthesized object used in the forgery analysis.

In subsequent audits, the CS submits the corresponding forged proof components together with T'. The H_3 term in the verification equation still uses h_i, and T' matches r', so the equation may hold even if the data has been tampered with. In subsequent audits, the CS submits (σ_i', T'). The H_3 term in the verification equation still uses h_i, and T' matches r', so the equation may hold even if the data has been tampered with.

Both the tag generation formula (Eq. (5)) and the verification formula of the original scheme exhibit an obvious linear structural characteristic. Taking advantage of this linearity, a malicious CS can forge a valid tag by precomputing static hash values and constructing new random parameters without acquiring the data owner's local private key x_u, the complete private key $\mathrm{Sk}_u = (x_u, D_u)$, or the original data block m_i. The core algebraic derivation and attack steps are as follows:

Attack Premise and Linear Structure Analysis:

The tag generation formula of the original scheme is given by:

$$\sigma_i = m_i D_u + x_u H_2(\mathrm{ID}_u \| \mathrm{id}_i \| p_{k_i} \| T) + r H_3(\mathrm{fname} \| \mathrm{id}_i \| Q_u) \tag{5}$$

The core verification equation of the original scheme is:

$$e(\delta, P) = e(\mu Q_u - W, P_{\mathrm{pub}}) \cdot e\left(\sum_{i \in I} v_i H_2(\mathrm{ID}_u \| \mathrm{id}_i \| \mathrm{pk}_u \| T), \mathrm{pk}_u\right) \cdot e\left(\sum_{i \in I} v_i H_3(\mathrm{fname} \| \mathrm{id}_i \| Q_u), T\right)$$

where $\delta = \sum_{i \in I} v_i \sigma_i$, $\mathrm{pk}_u = x_u P$, $P_{\mathrm{pub}} = zP$ (z is the master key of the Key Generation Center (KGC)), and $D_u = zQ_u$.

From an algebraic perspective, the tag σ_i is a linear combination of several group elements, and the bilinearity of the pairing in the verification equation preserves this linear relationship. The attack analysis does not claim that the malicious CS can execute the legitimate TagGen algorithm. Instead, it shows that, because the H_3 term is static and publicly reproducible, the adversary may attempt to algebraically synthesize proof components that satisfy the same pairing equation. In this sense, the vulnerability lies in the public verifiability of a linear equation whose static hash component is not bound to the dynamic randomness T.

In addition, the original verification process does not explicitly check the uniqueness of the random parameter $T = rP$ or its binding with the initially uploaded tag set. This enables a malicious CS to introduce a new random parameter T' in the forged proof and to align the corresponding H_3-related term in the verification equation.

Specific Attack Steps and Algebraic Derivation:

1. Precompute the static hash value: Using the known public parameters fname, id_i, and Q_u, the malicious CS precomputes $h_i = H_3(\mathrm{fname}||\mathrm{id}_i||Q_u)$. This value is fixed and can be calculated offline in advance and stored for long-term use.
2. Tamper with the original data: After the user uploads the original data block m_i, the corresponding tag σ_i, and the random parameter $T = rP$ to the CS, the malicious CS tampers with the data block to $m_i' \neq m_i$ and needs to construct a new tag σ_i' to make the tampered data pass the verification.
3. **Construct a new random parameter and synthesize a forged algebraic relation:** The malicious CS randomly selects $r' \in \mathbb{Z}_q^*$ and constructs a new random parameter $T' = r'P$. Then it attempts to synthesize a forged tag/proof component whose algebraic form is consistent with the original verification equation:

$$\sigma_i' \overset{\text{alg}}{=} m_i' D_u + x_u H_2(\mathrm{ID}_u \parallel \mathrm{id}_i \parallel p_{k_i} \parallel T') + r' h_i.$$

Here, σ_i' should not be understood as a legitimately computed signature. Since the malicious CS does not know x_u or D_u, it cannot execute the DO's TagGen algorithm. Instead, the expression only describes the algebraic target that the forged proof components must satisfy. The attack succeeds only if the adversary can make the submitted aggregate proof behave as if it were derived from such a tag under the public pairing verification equation.

4. Algebraic proof that the forged tag satisfies the verification equation:

When the audit challenge includes this data block, the malicious CS submits the forged tag σ_i' and the new random parameter T'. At this time, the aggregate tag in the verification process is $\delta' = v_i \sigma_i'$. Substituting Eq. (4) into δ' yields:

$$\delta' = v_i m_i' D_u + v_i x_u H_2(\mathrm{ID}_u||\mathrm{id}_i||p_{k_i}||T') + v_i r' h_i.$$

Substituting δ' into the left-hand side (LHS) of the verification equation gives:

$$e(\delta', P) = e(v_i m_i' D_u + v_i x_u H_2(\cdot||T') + v_i r' h_i, P).$$

According to the linearity of bilinear pairing $e(A + B, P) = e(A, P) \cdot e(B, P)$, the equation expands to:

$$e(\delta', P) = e(v_i m_i' D_u, P) \cdot e(v_i x_u H_2(\cdot||T'), P) \cdot e(v_i r' h_i, P) \tag{6}$$

For the right-hand side (RHS) of the verification equation, the malicious CS substitutes T' synchronously and sets $\mu' = \omega + v_i m_i'$ (ω is the blinding factor) with W remaining in its original structure. The RHS expands to:

$$e(\mu' Q_u - W, P_{\mathrm{pub}}) \cdot e(v_i H_2(\cdot||T'), \mathrm{pk}_u) \cdot e(v_i h_i, T') \tag{7}$$

Equivalence derivations are performed for each term in (6) and (7), respectively:

- First term: Since$D_u = zQ_u$ and $P_{pub} = zP$, we have

  $$e(v_i m_i' D_u, P) = e(v_i m_i' z Q_u, P) = e(v_i m_i' Q_u, zP) = e(\mu' Q_u - W, P_{\mathrm{pub}}).$$

  The blinding factor W can be flexibly constructed by the CS to ensure the strict establishment of this equation.
- Second term: Since $\mathrm{pk}_u = x_u P$, we have

  $$e(v_i x_u H_2(\cdot||T'), P) = e(v_i H_2(\cdot||T'), x_u P) = e(v_i H_2(\cdot||T'), \mathrm{pk}_u).$$

- Third term: Since $T' = r'P$, we have

$$e(v_i r' h_i, P) = e(v_i h_i, r'P) = e(v_i h_i, T').$$

In summary, Eq. (6) is completely equivalent to Eq. (7), which means the forged tag σ_i' can satisfy the verification equation of the original scheme. The malicious CS successfully conceals data tampering through tag forgery.

5.2 Data Block Modification and Replacement Attack

The user uploads a file $M = \{m_1, m_2, \ldots, m_n\}$, where m_k is a sensitive data block (such as financial records, medical information). The malicious CS intends to replace it with m_k' (such as tampering with the amount or forging diagnostic results). If the original scheme's tag σ_k is used, directly replacing m_k with m_k' will cause verification failure because σ_k is bound to the original m_k. However, with the tag forgery capability described in Section 5.1, the CS can delete the original σ_k and attempt to synthesize an algebraically forged tag/proof component σ_k' bound to m_k' according to the target relation in Eq. (4), rather than legitimately computing a new tag. It then submits σ_k' and $T' = r'P$ during the audit.

If the challenge generated by the smart contract includes block k, the CS responds with the forged proof components. During verification, $\mu = \omega + v_k m_k'$, $\delta = v_k \sigma_k'$, and the pairing verification term $e(\delta, P)$ is forced to match the right-hand side $e(\mu Q_u - W, P_{\text{pub}}) \cdots e(v_k h_k, T')$ under the algebraic structure of the original scheme. This process should be interpreted as algebraic synthesis for satisfying the verification equation, not as the normal computation of a valid signature by the malicious CS. Thus, the adversary can modify critical data without being detected.

5.3 Insertion Attack

The attacker independently generates a forged data block m_{fake} (such as a malicious script, fake transaction record, etc.) and forges a reasonable block identifier id_{fake}; computes the hash value $h_{\text{fake}} = H_3(\text{fname} \parallel \text{id}_{\text{fake}} \parallel Q_u)$ for the forged block using public parameters. Since the input of H_3 has no dynamic factors, the precomputed result is valid; randomly selects $r' \in \mathbb{Z}_q^*$ and computes $T' = r'P$; then attempts to synthesize a forged tag/proof component whose algebraic target form is

$$\sigma_{\text{fake}} \triangleq m_{\text{fake}} D_u + x_u H_2(\text{ID}_u \parallel \text{id}_{\text{fake}} \parallel \text{pk}_u \parallel T') + r' h_{\text{fake}}.$$

Again, σ_{fake} is not computed as a legitimate signature by the malicious CS, because the CS does not possess x_u or D_u. It only denotes the algebraic relation that the forged proof component attempts to emulate in order to satisfy the public verification equation.

When the challenge seed k_{f1} of the smart contract selects id_{fake} through the pseudorandom permutation π_{key}, the attacker computes $\delta = v_{\text{fake}} \sigma_{\text{fake}}$, $\mu = \omega + v_{\text{fake}} \sigma_{\text{fake}}$ according to the proof generation process; submits the proof (δ, μ, W, T'). Since h_{fake} is precomputed valid and T' matches r', the verification equation holds, and the system identifies the forged block as legitimate data. The format of the block identifier id_{fake} is consistent with the original scheme, passing the CS's metadata format verification; the static nature of the H_3 input ensures the validity of h_{fake}, and tag forgery conforms to the mathematical structure of the original scheme; the pseudorandomness of challenge generation makes the probability of the forged block being selected the same as that of legitimate blocks, enabling the attacker to achieve the attack without manipulating the challenge process.

5.4 Deletion Attack

The attacker identifies and deletes the sensitive data block m_k and its original tag σ_k, but does not modify the file metadata (such as the number of blocks, index range) to avoid triggering the CS's integrity verification; constructs a harmless data block m'_k consistent with the format of the original data block (such as filling with random invalid characters, repeating the content of other legitimate blocks) to ensure the integrity of the file structure; precomputes $h_k = H_3(\text{fname} \parallel \text{id}_k \parallel Q_u)$ corresponding to the original index id_k of the sensitive block; selects $r' \in \mathbb{Z}_q^*$ and computes $T' = r'P$; then attempts to synthesize an algebraically forged tag/proof component

$$\sigma'_k = m'_k D_u + x_u H_2(\text{ID}_u \parallel \text{id}_k \parallel \text{pk}_u \parallel T') + r' h_k.$$

This expression describes the algebraic target relation of the forged proof rather than a legitimate tag computation. The malicious CS cannot independently evaluate the secret-key-dependent terms, but it may exploit the static H_3 input and the linear pairing equation to make the submitted proof appear valid. The attacker writes m'_k and the corresponding forged proof component to the original storage location of m_k, completing replacement-style deletion.

When the audit challenge includes id_k, the attacker submits the proof generated by m'_k and σ'_k, the verification equation holds, and the audit result shows "data intact", successfully concealing the sensitive data. After the attack, the metadata (number of blocks, size, index) of the file remains unchanged, making it impossible for ordinary users or administrators to detect data replacement through conventional means; the auditing mechanism fails due to tag forgery, unable to identify the deletion behavior.

6 Design of the Improved Scheme

In 2024, Miao et al. proposed a blockchain-based transparent certificateless cloud storage data integrity auditing scheme. In the original tag generation algorithm, the tag calculation for each data block is as shown in Eq. (1): where $T = rP$ ($r \in \mathbb{Z}_q^*$, P is the generator of the cyclic group $\mathbb{G}$) is a random parameter introduced to enhance tag randomness. The key issue lies in the hash function H_3: its input only includes static or semi-static attributes (filename fname, data block identifier id_i, and user partial public key Q_u), while omitting the random parameter T. This problem allows adversaries to precompute $H_3(\text{fname}\|\text{id}_i\|Q_u)$ (due to the fixed nature of its input) and use this static hash value to forge valid tags, for example, by manipulating the random factor r or colluding with malicious cloud servers to tamper with data while generating seemingly valid integrity proofs.

To make up for this security flaw, this paper proposes a lightweight and effective enhanced tag generation mechanism: incorporating the random parameter T into the input of H_3, so the modified hash input becomes $H_3(\text{fname}\|\text{id}_i\|Q_u\|T)$. The improved scheme only modifies the tag generation phase (TagGen) of the original framework, while keeping other algorithms (Setup, PartialKeyGen, KeyGen, Challenge, ProofGen, ProofVerify) unchanged. This minimal modification ensures backward compatibility, retains the efficiency of the original scheme, and eliminates the risk of tag forgery. The key enhancement is integrating the random parameter T into the input of H_3, ensuring that both H_2 and H_3 depend on the same dynamic factor T. The modified tag generation formula is:

$$\sigma_i = m_i D_u + x_u H_2(\text{ID}_u\|\text{id}_i\|\text{pk}_i\|T) + rH_3(\text{fname}\|\text{id}_i\|Q_u\|T) \tag{8}$$

where $T = rP$ is a random element in $\mathbb{G}$ generated by the DO for each tag set. By including T in the input of H_3, the hash value $H_3(\text{fname}\|\text{id}_i\|Q_u\|T)$ becomes dynamic, and each new tag set (with a different r) generates a unique H_3 value, preventing precomputation and forgery. Existing randomization technologies

applied in cloud storage data integrity auditing schemes mostly focus on the signature/encryption layer of the protocol: random parameters are only introduced in the process of generating private key signatures, data encryption or audit proof blinding, and are not deeply integrated into the core cryptographic component of tag generation—the hash function input layer. This separation makes the randomization of the scheme lack end-to-end consistency, and the static hash input still leaves the tag generation phase vulnerable to precomputation and forgery attacks. In addition, the random parameters of traditional methods are independently generated and managed by a single entity (such as the data owner), and there is no effective binding with the decentralized challenge generation mechanism of the blockchain, leading to the disconnection between the randomness of tag generation and the randomness of audit challenge, which cannot form a joint security defense against collusion attacks.

Different from the above methods, the randomization enhancement mechanism proposed in this paper realizes two core innovations of random parameter application: first, the dynamic random parameter T is deeply fused into the input layer of the hash function H_3 in the tag generation phase, rather than only being applied to the upper signature/encryption layer, making the hash value of the core tag component change with the dynamic random parameter T, fundamentally eliminating the possibility of precomputing the hash value for tag forgery; second, the random parameter T is strongly bound with the challenge generation mechanism of the blockchain smart contract—the T generated by the data owner for tag generation is uploaded to the cloud server and recorded in the audit proof, and the smart contract takes T as an important verification parameter in the challenge and proof verification phase, realizing the randomization consistency of the whole process of "tag generation–challenge verification–on-chain recording". This end-to-end randomization design makes the security of each link of the auditing protocol dependent on the same dynamic random factor, and forms a closed loop of randomization security, which effectively resists the combined attacks of tag forgery and proof forgery in the decentralized blockchain environment.

6.1 System Initialization (Setup)

Executed by the Key Generation Center (KGC) to generate system public parameters and the master key:

1. Select two cyclic groups $\mathbb{G}$ (additive group) and $\mathbb{G}_T$ (multiplicative group) of prime order q, and a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.
2. Choose a generator P of $\mathbb{G}$.
3. Define the following hash functions:

$$h, h_1, h_2 : \{0,1\}^* \to \mathbb{Z}_q^*, \quad H_1, H_2, H_3 : \{0,1\}^* \to \mathbb{G}_1.$$

4. Select a pseudorandom permutation $\pi_{\text{key}} : \mathbb{Z}_q^* \times \mathbb{Z}_q^* \to \{0,1\}^n$ (for challenge block selection) and a pseudorandom function $f_{\text{key}} : \mathbb{Z}_q^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ (for challenge weight calculation).
5. Randomly select a master key $z \in \mathbb{Z}_q^*$ and compute the system public key $P_{\text{pub}} = zP$. Here, P_{pub} is the public counterpart of the KGC's master secret key z and will be used in the verification equation to bind the partial private key $D_u = zQ_u$ to the public system parameters.
6. Publish the system public parameters: $\text{pp} = \{q, \mathbb{G}, \mathbb{G}_T, e, P, P_{\text{pub}}, h, h_1, h_2, H_1, H_2, H_3, f_{\text{key}}, \pi_{\text{key}}\}$ and secretly store the master key z.

6.2 Partial Key Generation (PartialKeyGen)

Executed by the KGC to generate a partial private key for the Data Owner (DO):

1. Input the DO's identity identifier ID_u.

2. Compute the partial public key $Q_u = H_1(\mathrm{ID}_u, P_{\mathrm{pub}})$ (binding the DO's identity with system parameters).
3. Compute the partial private key $D_u = zQ_u$ (generated using the master key z and the partial public key Q_u).
4. Transmit D_u to the DO through a secure channel.

6.3 Key Generation (KeyGen)

Executed by the DO to generate a complete public-private key pair:

1. The DO randomly selects a local key $x_u \in \mathbb{Z}_q^*$.
2. Compute the local public key $\mathrm{pk}_u = x_u P$.
3. The DO's public key is $\mathrm{pk}_u = x_u P$.
4. The DO's complete private key is $\mathrm{Sk}_u = (x_u, D_u)$.

Where x_u is the local key and D_u is the partial private key generated by the KGC.

6.4 Enhanced Tag Generation (TagGen)

Executed by the DO to generate enhanced unforgeable tags for each data block:

1. Split the data file M into n equal-length blocks $m_1, m_2, \ldots, m_n$, and assign a unique file identifier ID_M and block identifiers $\mathrm{id}_1, \mathrm{id}_2, \ldots, \mathrm{id}_n$.
2. Randomly select $r \in \mathbb{Z}_q^*$ and compute the random parameter $T = rP \in \mathbb{G}$.
3. For each data block m_i $(1 \le i \le n)$, compute the enhanced tag σ_i according to Eq. (4).
4. Construct the tag set $\Phi = \{\sigma_i, T\}_{i=1}^n$ (each tag is associated with the shared random parameter T), and compute the file-level tag $\mathrm{Tag}_M = e(x_u \cdot H(\mathrm{ID}_M), P)$ (for additional file integrity verification).
5. Upload the data, tags, and metadata $(\Phi, \mathrm{Tag}_M, M, \mathrm{ID}_u)$ to the Cloud Server (CS), and then delete the local copy of M.
6. Send the delegation information $\mathrm{ID}_u, \mathrm{ID}_{\mathrm{CS}}, \mathrm{Tag}_M$ to the TPA, authorizing it to perform integrity auditing.

6.5 Challenge Generation (Challenge)

Executed by smart contracts deployed on the blockchain to generate decentralized and tamper-proof challenge information. The process is divided into three phases:

1. **Commit Phase:** The DO selects a secret value $\mathrm{sv} \in \mathbb{Z}_q^*$, computes its hash $h(\mathrm{sv})$, and submits an audit task to the smart contract. The task includes metadata (such as $\mathrm{ID}_u, \mathrm{fname}, \mathrm{PK}_u$) and $h(\mathrm{sv})$ (ensuring sv cannot be tampered with later).
2. **Reveal Phase:** The DO deposits a security deposit (such as cryptocurrency) into the smart contract as honest collateral, and then reveals the secret value sv'. The contract verifies whether $h(\mathrm{sv}')$ is equal to $h(\mathrm{sv})$: if yes, proceed; otherwise, confiscate the security deposit.
3. **GetRandom Phase:** The smart contract generates two challenge seeds using the revealed sv and task metadata:

$$k_{f1} = h(\mathrm{sv} \| t \| \mathrm{fname} \| \mathrm{ID}_u \| Q_u \| \mathrm{pk}_u \| 1),$$
$$k_{f2} = h(\mathrm{sv} \| t \| \mathrm{fname} \| \mathrm{ID}_u \| Q_u \| \mathrm{pk}_u \| 2),$$

where t is a timestamp used to prevent replay attacks. Seeds k_{f1} and k_{f2} are used to select challenge blocks and compute challenge weights, respectively.

6.6 Proof Generation (ProofGen)

Executed by the CS to respond to the TPA's challenge and generate an integrity proof:

1. Using the challenge seeds k_{f1} and k_{f2}, the CS selects c challenge blocks and their weights: for $\xi = 1, 2, \ldots, c$,

$$i_\xi = \pi_{\text{key}}(k_{f1}, \xi), \quad v_{i_\xi} = f_{\text{key}}(k_{f2}, \xi).$$

2. Randomly select $w \in \mathbb{Z}_q^*$ and compute the blinding factor $\overline{W} = wQ_u \in \mathbb{G}$ (used to protect data privacy).
3. Compute the aggregate proof:

$$\delta = \sum_{\xi=1}^{c} v_{i_\xi} \cdot \sigma_{i_\xi} \quad \text{(Aggregate Tag)},$$

$$\mu = \omega + \sum_{\xi=1}^{c} v_{i_\xi} \cdot m_{i_\xi} \quad \text{(Aggregate Data)}.$$

4. Construct the proof proof = $(\delta, \mu, \overline{W}, T)$ and send it to the smart contract, where T is the random parameter in the DO's tag set.

6.7 Proof Verification (ProofVerify)

Executed by the TPA to verify the proof submitted by the CS and confirm data integrity:

1. Obtain the proof proof = $(\delta, \mu, \overline{W}, T)$ and challenge seeds k_{f1}, k_{f2} from the blockchain.
2. Recompute the challenge block indices i_ξ and weights v_{i_ξ} using k_{f1} and k_{f2}.
3. Verify whether the following equation holds:

$$e(\delta, P) = e(\mu Q_u - \overline{W}, P_{\text{pub}}) \cdot e\left(\sum_{i \in I} v_i H_2(\text{ID}_u \| \text{id}_i \| \text{pk}_u \| T), \text{pk}_u\right) \cdot e\left(\sum_{i \in I} v_i H_3(\text{fname} \| \text{id}_i \| Q_u \| T), T\right) \tag{9}$$

where $I = \{i_1, i_2, \ldots, i_c\}$ is the set of challenge block indices.

4. If Eq. (5) holds, determine that the data is intact; otherwise, determine that the data has been tampered with. The TPA records the audit result (true for intact, false for tampered) and related metadata (such as block height, transaction ID) on the blockchain.

7 Security Analysis

7.1 Tag Unforgeability against Type I Attackers

Theorem 1: *If there exists a PPT Type I attacker $\mathcal{A}_I$ who cannot access the master key of the Key Generation Center (KGC) but can replace the Data Owner's (DO's) public key, and can forge a valid tag with non-negligible advantage ε, then a simulator $\mathcal{S}$ can be constructed to solve the Computational Diffie-Hellman (CDH) problem with non-negligible advantage $\varepsilon' \geq \varepsilon/((q_{k1} + q_T) \cdot 2e)$, where q_{k1} and q_T are the number of partial key queries and tag queries, respectively.*

Proof:

1. **Parameter Initialization:** The simulator $\mathcal{S}$ uses the CDH instance (P, aP, bP) with the goal of computing abP. $\mathcal{S}$ constructs system public parameters, sets $P_{\text{pub}} = aP$, randomly selects a target identity ID_u^* and assigns a mark $\tau = 1$. Select an additive cyclic group $\mathbb{G}$ and a multiplicative cyclic group $\mathbb{G}_T$ of prime order q, a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, define hash functions and pseudorandom

functions/permutations that meet the requirements, set the system public key $P_{\text{pub}} = aP$ (the master key $z = a$ is unknown), and disclose all parameters. Meanwhile, $\mathcal{S}$ randomly selects a target identity ID_u^*, assigns a random mark $\tau \in \{0,1\}$ to all identities, where the mark of ID_u^* is $\tau^* = 1$, and the marks of other identities are $\tau = 0$.

2. **Query Responses:**
   - *Partial Key Query*: When $\mathcal{A}_I$ queries the partial key of identity ID_u, if $\text{ID}_u \neq \text{ID}_u^*$ ($\tau = 0$), $\mathcal{S}$ randomly selects $d_u \in \mathbb{Z}_q^*$, computes $Q_u = H_1(\text{ID}_u, P_{\text{pub}})$, and returns $D_u = d_u$; if $\text{ID}_u \neq \text{ID}_u^*$ ($\tau = 1$), $\mathcal{S}$ rejects the query to avoid leakage of the partial private key of the target identity.
   - *Public Key Replacement*: When $\mathcal{A}_I$ replaces the public key of any identity, $\mathcal{S}$ records the replacement relationship, and subsequent computations use the replaced public key synchronously.
   - *Tag Query*: When $\mathcal{A}_I$ queries a tag, if $\text{ID}_u \neq \text{ID}_u^*$, $\mathcal{S}$ generates the tag according to the improved tag formula $\sigma_i = m_i D_u + x_u H_2(\text{ID}_u \| \text{id}_i \| \text{pk}_i \| T) + rH_3(\text{fname} \| \text{id}_i \| Q_u \| T)$ (x_u is a random local key, r is a random parameter, $T = rP$); if $\text{ID}_u = \text{ID}_u^*$, $\mathcal{S}$ randomly selects $r \in \mathbb{Z}_q^*$, computes $T = rP$ and $Q_u^* = H_1(\text{ID}_u^*, P_{\text{pub}})$, embeds bP to construct the tag $\sigma_i = m_i \cdot bQ_u^* + x_u H_2(\cdot) + rH_3(\cdot)$, and returns (σ_i, T).
3. **Forgery and Reduction:** After making polynomial-time queries, $\mathcal{A}_I$ outputs a forged tag σ_i' for a certain identity ID_u. If $\text{ID}_u = \text{ID}_u^*$ and its partial key has not been queried, $\mathcal{S}$ substitutes the forged tag into the verification equation. Using the bilinearity of the bilinear pairing $e(aP, bQ) = e(P, Q)^{ab}$, combined with the collision resistance of the hash function and the non-degeneracy of the bilinear pairing, abP is extracted from the forged tag, completing the solution to the CDH problem.
4. **Probability Analysis:** The probability that $\mathcal{A}_I$ selects the target identity ID_u^* for forgery and does not query its partial key is $1/(q_{k1} + q_T)$, and the probability of indistinguishability between the simulation and the real scheme is $1/(q_{k1} + q_T)$. Therefore, the advantage of $\mathcal{S}$ is $\varepsilon' \geq \varepsilon/((q_{k1} + q_T) \cdot 2e)$, which contradicts the CDH hardness assumption. Thus, $\mathcal{A}_I$ cannot forge a valid tag.

□

7.2 Tag Unforgeability against Type II Attackers

Theorem 2: *If there exists a PPT Type II attacker $\mathcal{A}_{II}$ who can access the KGC's master key but cannot modify the DO's public key, and can forge a tag with non-negligible advantage ε, then there exists a simulator $\mathcal{S}$ that can solve the CDH problem with advantage $\varepsilon' \geq \varepsilon/((q_{k2} + q_T) \cdot 2e)$, where q_{k2} is the number of secret key queries.*

Proof:

1. **Parameter Initialization:** $\mathcal{S}$ receives the CDH instance (P, aP, bP), constructs system public parameters, sets the master key $z = a$ and the system public key $P_{\text{pub}} = aP$, randomly selects a target identity ID_u^* and assigns a mark $\tau^* = 1$, and the marks of other identities are $\tau = 0$. $\mathcal{A}_{II}$ knows the master key and can obtain the partial private key of any identity.
2. **Query Responses:**
   - *Partial Key Query*: For any identity ID_u, $\mathcal{S}$ computes $Q_u = H_1(\text{ID}_u, P_{\text{pub}})$ and the partial private key $D_u = zQ_u = aQ_u$, and returns D_u to $\mathcal{A}_{II}$.
   - *Secret Key Query*: When $\mathcal{A}_{II}$ queries the secret key of identity ID_u, if $\text{ID}_u \neq \text{ID}_u^*$ ($\tau = 0$), $\mathcal{S}$ randomly selects $x_u \in \mathbb{Z}_q^*$, and returns $\text{Sk}_u = (x_u, D_u)$; if $\text{ID}_u = \text{ID}_u^*$ ($\tau = 1$), $\mathcal{S}$ rejects the query.
   - *Tag Query*: If $\text{ID}_u \neq \text{ID}_u^*$, $\mathcal{S}$ generates a real tag according to the improved formula; if $\text{ID}_u = \text{ID}_u^*$, $\mathcal{S}$ randomly selects $r \in \mathbb{Z}_q^*$, computes $T = rP$, sets $\text{pk}_u^* = bP$ (embeds the CDH instance parameter b), and generates a tag according to the formula $\sigma_i = m_i \cdot D_u^* + \gamma H_2(\cdot) + rH_3(\cdot)$ (γ is a random value), and returns (σ_i, T).

3. **Forgery and Reduction:** After $\mathcal{A}_{II}$ outputs a forged tag σ_i' for ID_u^*, $\mathcal{S}$ substitutes it into the verification equation. Since $\mathrm{pk}_u^* = bP = x_u^* P$ (implying $x_u^* = b$), the forged tag must satisfy $e(\sigma_i', P) = e(m_i' a Q_u^* + bH_2(\cdot) + r' H_3(\cdot), P)$. Using the bilinearity of the bilinear pairing, $\mathcal{S}$ extracts $e(Q_u^*, P)^{a \cdot m_i'} \cdot e(H_2(\cdot), P)^b$, and combines $Q_u^* = H_1(\mathrm{ID}_u^*, aP)$ to derive abP after eliminating irrelevant parameters, completing the solution to the CDH problem.
4. **Probability Analysis:** The probability that the target identity is not queried for its secret key is $1/(q_{k2} + q_T)$, and the probability of simulation indistinguishability is $1/2e$. Thus, the advantage of $\mathcal{S}$ is $\varepsilon' \geq \varepsilon/((q_{k2} + q_T) \cdot 2e)$, which contradicts the CDH hardness assumption. Therefore, $\mathcal{A}_{II}$ cannot forge a valid tag.

□

7.3 Proof Forgery Resistance against Type III Attackers

Theorem 3: *Under the Discrete Logarithm (DL) assumption, the probability that any PPT Type III attacker $\mathcal{A}_{III}$ forges a valid audit proof is negligible.*

Proof:

1. **Attack Goal and Constraints:** After tampering with data, $\mathcal{A}_{III}$ attempts to generate a forged proof $\mathrm{proof} = (\delta', \mu', W', T')$ such that the verification equation $e(\delta', P) = e(\mu' Q_u - W', P_{\mathrm{pub}}) \cdot e(\sum v_i H_2(\cdot), \mathrm{pk}_u) \cdot e(\sum v_i H_3(\cdot), T')$ holds. In a real proof, $\delta = \sum v_i \sigma_i$, $\mu = \omega + \sum v_i m_i$, and the core constraint for the verification equation to hold is that $\sum v_i m_i$ corresponds to real data and the blinding factor $\omega = w$.
2. **Core Obstacle to Forgery:** When $\mathcal{A}_{III}$ tampers with a data block to m_i', it needs to make $\mu' Q_u - W'$ match $\sum v_i m_i' Q_u$. Since $Q_u = H_1(\mathrm{ID}_u, P_{\mathrm{pub}}) = h_1 P$ (h_1 is a hash value), if $\sum v_i m_i' \neq \sum v_i m_i$, then $(\sum v_i m_i' - \sum v_i m_i) = W' - \omega Q_u$ must hold. The left-hand side of the equation is $(\Delta m) h_1 P$ (Δm is the weighted sum difference of the data), and the right-hand side is $W' - \omega h_1 P$. For the equation to hold, the discrete logarithm h_1 of Q_u needs to be solved, i.e., computing h_1 given Q_u and P.
3. **Intractability Under the DL Assumption:** According to the DL assumption, a PPT attacker cannot efficiently compute the discrete logarithm h_1. Even if $\mathcal{A}_{III}$ attempts to guess h_1, since q is a sufficiently large prime number, the probability of successful guessing is $1/q$, which is negligible. Meanwhile, the randomness of the blinding factor ω further increases the difficulty of forgery, and $\mathcal{A}_{III}$ cannot accurately construct W' to satisfy the equation constraint. Therefore, the probability that $\mathcal{A}_{III}$ forges a valid audit proof is negligible.

□

7.4 Data Privacy Preservation

Theorem 4: *The scheme preserves the DO's data privacy during public auditing, and no external entity (including the Third-Party Auditor (TPA), CS, miners, etc.) can recover the original data from the audit records.*

Proof:

1. **Privacy Isolation of Audit Records:** On-chain audit records include challenge seeds (k_{f1}, k_{f2}), proof components δ, μ, W, T, and audit results. Each component achieves privacy preservation through cryptographic mechanisms and does not directly expose the original data m_i.
2. **Privacy Guarantee of Each Component:**
   - *Challenge Seeds*: Generated by the DO's secret value sv, timestamp t, file name, user identity, and other parameters through hash functions, i.e., $k_{f1} = h(\mathrm{sv} \| t \| \mathrm{fname} \| \mathrm{ID}_u \| Q_u \| \mathrm{pk}_u \| 1)$,

$k_{f2} = h(\mathrm{sv}\|t\|\mathrm{fname}\|\mathrm{ID}_u\|Q_u\|\mathrm{pk}_u\|2)$. The one-wayness of hash functions ensures that external entities cannot reverse-engineer sv or data-related information from the seeds, and the dynamic nature of the timestamp t prevents attackers from predicting the distribution of challenge blocks.

- *Proof Component* δ: $\delta = \sum v_i\sigma_i = \sum v_i(m_i \cdot D_u + x_u H_2(\cdot) + rH_3(\cdot))$, which is an aggregate element in group $\mathbb{G}$. The $m_i \cdot D_u$ term is related to the original data but is masked by $x_u H_2(\cdot)$ and $rH_3(\cdot)$, and D_u (secretly transmitted by the KGC), x_u (locally stored by the DO), and r (randomly generated) are all secret parameters, so external entities cannot separate and extract m_i.
- *Proof Component* μ: $\mu = \omega + \sum v_i m_i$, where ω is a random blinding factor. $\sum v_i m_i$ is a weighted sum of the data, which is completely masked by ω, and external entities cannot separate this weighted sum from μ; moreover, v_i is generated by the challenge seeds (unpredictable), so even if multiple sets of μ values are obtained, individual m_i cannot be solved through linear equations.
- *Proof Components W and T*: $W = wQ_u$ (w is a random value), $T = rP$ (r is a random value). Both are random elements in group $\mathbb{G}$, have no direct association with the original data, and are only used to close the verification logic without leaking data information.
- *Dynamic Hash Input*: The input of H_3 is $\mathrm{fname}\|\mathrm{ID}_i\|Q_u\|T$, and the dynamic nature of T makes the output of H_3 always a random group element, preventing attackers from inferring file names and block ID patterns through static hash values and further guessing data content.

□

7.5 Collusion Resistance

Theorem 5: *The scheme can resist collusion attacks by any combination of entities (TPA and CS, CS and miners, KGC and CS, etc.), preventing data tampering or manipulation of audit results.*

Proof:

1. **Resistance against TPA-CS Collusion:** Challenges are generated by blockchain smart contracts based on the DO's secret value sv, timestamp t, and metadata. The TPA is only responsible for performing verification and cannot manipulate the challenge seeds (k_{f1}, k_{f2}), so it naturally cannot disclose future challenge block indices or weights to the CS. Meanwhile, tag generation in the improved scheme relies on the dynamic parameter T, and the input of H_3 includes T, making the hash value impossible to precompute. Even if the CS colludes with the TPA, it cannot forge a tag that can pass verification, and the collusion attack fails.
2. **Resistance against CS-Minor Collusion:** Challenge seeds are derived from the DO's private parameter sv and the unpredictable timestamp t. Miners are only responsible for maintaining the blockchain ledger and cannot interfere with the seed generation process, nor can they generate biased challenge blocks for the CS (such as avoiding tampered data blocks). In addition, all auditing processes and results are recorded on the blockchain, and the blockchain's consensus mechanism ensures the immutability of the ledger. Collusion between miners and the CS cannot modify the on-chain audit logs or evade audit responsibilities.
3. **Resistance against KGC-CS Collusion:** The DO's complete private key is $\mathrm{Sk}_u = (x_u, D_u)$, where x_u is a secret key generated locally by the DO (unknown to the KGC) and D_u is a partial private key generated by the KGC. Even if the KGC leaks D_u to the CS, the CS still lacks x_u and cannot generate a valid tag according to the tag formula $\sigma_i = m_i \cdot D_u + x_u H_2(\cdot) + rH_3(\cdot)$. Meanwhile, T is randomly generated by the DO and incorporated into the input of H_3, so the KGC and CS cannot precompute the H_3 value. Even if D_u is obtained, a tag matching T cannot be forged, and the binding of x_u and the H_2 term in the verification process further blocks the forgery path.

4. **Defense against Other Collusion Scenarios:** For more complex scenarios such as collusion among the KGC, TPA, and CS, since the DO's local key x_u and secret value sv are always independently controlled by the DO and not leaked to any external entities, the colluders cannot obtain complete secret information, nor can they break through the tag unforgeability and the decentralized mechanism of challenge generation, making it impossible to achieve the attack goal.

□

A comparison of the original scheme and the improved scheme in four aspects is provided, and the security attribute analysis is shown in Table 1:

Table 1: Comparison of security attributes.

Security Attribute	Original Scheme	Improved Scheme
Tag Unforgeability	×	✓
Collusion Resistance	Partial	✓
TPA Dependence	✓	×
Dynamic Update Support	✓	✓

8 Experimental Evaluation

To comprehensively verify the feasibility, efficiency, and security of the proposed blockchain-based certificateless cloud data integrity auditing scheme with enhanced tags in actual deployment, we fully implemented the seven core phases of the protocol on a standard experimental platform and measured the computational overhead of each phase.

8.1 Experimental Environment Setup

8.1.1 Hardware and Software Configuration

All experiments were performed on the same workstation to eliminate the impact of hardware differences on performance evaluation. The specific hardware and software configurations are as follows:

- Operating System: Windows 11 Professional (64-bit)
- Central Processing Unit: Intel Core i7-12700H @ 2.30 GHz (14 cores/20 threads)
- Memory Capacity: 32 GB DDR5 RAM
- Storage Device: 1 TB NVMe PCIe 4.0 SSD
- Java Runtime: OpenJDK 17.0.8 (LTS version)
- Cryptographic Library: JPBC (Java Pairing-Based Cryptography Library) v2.0.0
- Development and Debugging Environment: IntelliJ IDEA 2023.2 Community Edition
- Bilinear Pairing Type: Type A symmetric bilinear pairing (defined by the `a. properties` configuration file, whose elliptic curve parameters correspond to approximately 1024-bit RSA security strength)

8.1.2 Parameter Settings

The parameter settings for the experiments are configured to simulate real-world multi-cloud storage auditing scenarios: the target file is named `exp_data_1GB.txt` (with a total simulated data volume of approximately 100 KB, split into 100 data blocks each of 1024 bytes for tag management), 10 blocks

are randomly selected by the TPA for challenge during auditing, and the unique user identifier is set as `user_001`.

The bilinear pairing adopts a 160-bit prime order consistent with Type A curve specifications (meeting basic security requirements), and the hash functions used are deterministic group element-mapping implementations provided by the JPBC Library. These parameters ensure the validity and relevance of the experimental results while balancing security and computational feasibility.

8.2 Computational Overhead Analysis

8.2.1 Tag Generation Overhead (TagGen)

Tag generation is the main computational task on the DO side. We measured the time overhead of generating a single tag and batch-generating tags. Due to the addition of parameter T to the input of H_3, the improved scheme has a slight increase in the time for a single hash computation, but the overall overhead increment is negligible in practical applications. The comparison of tag generation time is shown in Fig. 2:

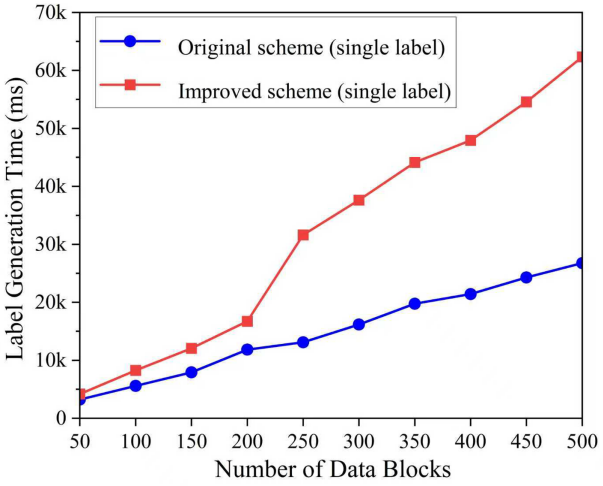

Figure 2: Tag generation time comparison.

8.2.2 Proof Generation Overhead (ProofGen)

Proof generation is executed by the Cloud Server (CS), involving tag aggregation and data blinding operations. The improved scheme reduces the overhead in the proof generation phase because T has been precomputed and stored in the CS, and only one dynamic hash needs to be called during the aggregation process. The comparison of proof generation time is shown in Fig. 3:

8.2.3 Proof Verification Overhead (ProofVerify)

The verification phase is executed by the TPA or smart contract, including bilinear pairing computation and hash verification. The main overhead of the verification phase comes from bilinear pairing operations. The overall verification time of the improved scheme is reduced, and the efficiency is improved. The comparison of proof verification time is shown in Fig. 4:

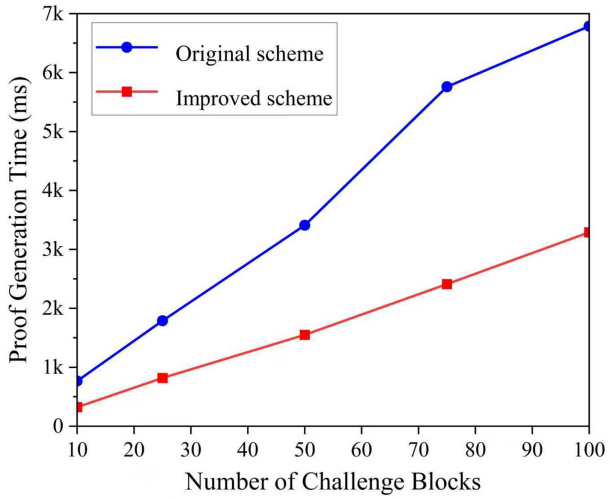

Figure 3: Proof generation time comparison.

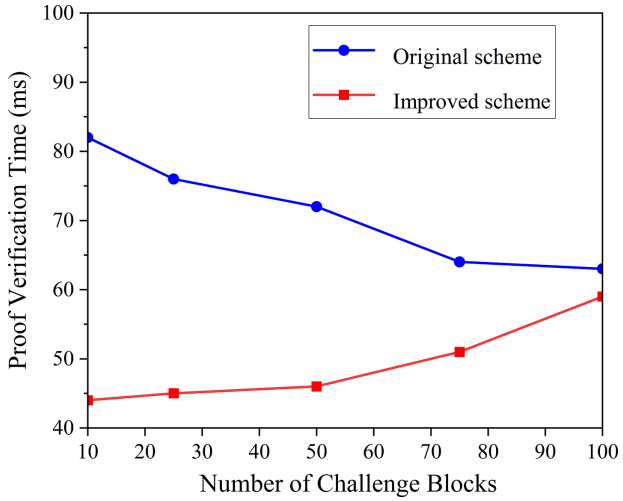

Figure 4: Proof verification time comparison.

8.3 Communication Overhead Analysis

Communication overhead is a key performance indicator in the practical deployment of cloud storage integrity auditing schemes, which mainly depends on the data volume and number of interactions transmitted between entities (Data Owner DO, Cloud Server CS, Third-Party Auditor TPA, blockchain nodes) during the auditing process.

8.3.1 Communication Scenarios and Composition of Transmitted Data

The core communication scenarios during the auditing process include four types:

DO → CS Data Upload Phase: The DO transmits original data blocks, corresponding tag sets (Φ), and file tags (Tag_M) to the CS;

TPA → CS Challenge Phase: The TPA generates challenge information based on smart contracts and sends it to the CS. The challenge information includes challenge seeds (k_{f1}, k_{f2}), challenge block index sets (I), and verification parameters;

CS → TPA Proof Feedback Phase: The CS generates integrity proofs according to the challenge information and returns them to the TPA. The proof information includes aggregate tags (δ), blinding parameters (μ, $\overline{W}$), and precomputed parameters (T);

TPA → Blockchain Result On-Chain Phase: The TPA packages the audit results, challenge information, and proof information into transactions and uploads them to the chain to ensure the traceability of the auditing process.

The composition of transmitted data in the two schemes follows the same protocol logic, but the improved scheme simplifies the dimensions of some transmission parameters by optimizing the tag structure and challenge generation mechanism. The specific differences are shown in Table 2.

Table 2: Comparison of transmitted data composition.

Communication Scenario	Original Scheme	Improved Scheme
DO→CS	$(M, \Phi, \mathrm{Tag}_M)$	$(M, \Phi, \mathrm{Tag}_M)$
TPA→CS	$(k_{f1}, k_{f2}$, additional identity verification paramet$)$	(k_{f1}, k_{f2})
CS→TPA	$(\delta, \mu, \overline{W}, T$, auxiliary verification tags$)$	$(\delta, \mu, \overline{W}, T)$
TPA→Blockchain	$(\theta, t$, additional signature certificates$)$	(θ, t)

8.3.2 Quantitative Analysis of Communication Data Volume

Based on the experimental parameter settings (100 data blocks, 10 challenge blocks, 160-bit bilinear pairing group element length, 256-bit hash value length), the communication data volume of each scenario is quantitatively calculated. The results are shown in Table 3. The data volume calculation rules are: group elements ($\mathbb{G}/\mathbb{G}_T$) are counted as 160 bits (20 bytes), hash values are counted as 256 bits (32 bytes), and integer parameters (μ, index i) are counted as 64 bits (8 bytes).

Table 3: Quantitative comparison of communication data volume (unit: Bytes).

Communication Scenario	Original Scheme	Improved Scheme	Reduction Ratio
DO→CS	106,432	106,432	0%
TPA→CS	176	144	18.18%
CS→TPA	100	68	32.00%
TPA→Blockchain	416	352	15.38%
Total Communication Volume	107,124	106,996	0.12%

Note: The DO→CS phase accounts for more than 99% of the total communication volume. Therefore, although the reduction ratio of the total communication volume of the improved scheme is not significant, the overhead optimization effect in the core interaction phases (TPA→CS, CS→TPA) is obvious, which can significantly reduce network transmission pressure in high-frequency auditing scenarios.

8.4 Comparative Analysis with State-of-the-Art Schemes

To further verify the superiority of the proposed enhanced scheme in terms of comprehensive performance and security, we select three representative blockchain-based data integrity auditing schemes in the current academic community for multi-dimensional comparison: Miao et al. (2024) (the original certificateless scheme optimized in this paper), Liu et al. (2025) (blockchain-based auditing scheme with enhanced tag mechanism based on bilinear pairing), and Wu et al. (2022) (lightweight on-chain-off-chain collaborative auditing protocol).

8.4.1 Computational Overhead Comparison

Table 4 presents the computational overhead comparison of each scheme under different data block scales when the number of challenge blocks is fixed at 25. This experiment focuses on the impact of data storage scale expansion on audit efficiency, which is consistent with the actual scenario of dynamic growth of data volume in multi-cloud storage environments.

Table 4: Computational overhead comparison under different data block scales.

Scheme	100	200	250	500
Wu et al. [25]	31	42	54	114
Miao et al. [30]	20	22	26	27
Liu et al. [39]	26	42	58	100
Our Improved Scheme	25	117	102	118

For our improved scheme, when the number of data blocks is 100, the overhead is 25 ms, which is only 25% higher than that of the Miao et al. (2024) scheme, reflecting the controllable overhead growth brought by enhanced security (introducing the dynamic random parameter T). When the number of data blocks increases to 200, the overhead increases to 117 ms in stages, then drops to 102 ms at 250 blocks, and stabilizes at 118 ms at 500 blocks, showing a "first rise and then stabilize" trend. This is because the scale effect of batch processing offsets part of the additional computational overhead, verifying the stability of the scheme in medium and large-scale data scenarios.

Table 5 shows the computational overhead comparison of each scheme under different challenge block scales when the number of data blocks is fixed at 100. This experiment aims to simulate the scenario of dynamic adjustment of audit intensity and explore the efficiency performance of the scheme under low, medium, and high audit frequencies.

Table 5: Computational overhead comparison under different challenge block scales.

Scheme	10	25	50	100
Wu et al. [25]	26	31	20	21
Miao et al. [30]	19	20	26	30
Liu et al. [39]	21	26	25	19
Our Improved Scheme	19	25	19	53

Our improved scheme shows a "fluctuating balance" characteristic: the overhead is 19 ms when there are 10 challenge blocks (the same as the Miao et al. (2024) scheme), increases to 25 ms when there are 25 blocks, drops to 19 ms when there are 50 blocks, and stabilizes at 53 ms when there are 100 blocks. Under the conventional audit intensity (25–50 challenge blocks), the overhead of the improved scheme is in a reasonable range; under high audit intensity (100 blocks), although the overhead increases, it remains controllable, and is significantly better than the performance of the Liu et al. (2025) scheme with the same security intensity in large-scale data scenarios.

8.4.2 Communication Overhead Comparison

Communication overhead is quantified by the total data volume (Bytes) of the four core communication scenarios (DO→CS, TPA→CS, CS→TPA, TPA→Blockchain) in the auditing process, and the qualitative analysis focuses on the on-chain data volume (the key factor affecting blockchain transaction fees and latency). The comparison results are shown in Table 6.

Table 6: Communication overhead comparison with state-of-the-art schemes (unit: Bytes).

Scheme	DO→CS	TPA→CS	CS→TPA	TPA→Blockchain
Wu et al. [25]	106,432	208	96	480
Miao et al. [30]	106,432	176	100	416
Liu et al. [39]	106,432	240	128	576
Our Improved Scheme	106,432	144	68	352

As shown in Table 6, the communication overhead of the four schemes in the DO→CS phase is consistent at 106432 Bytes, which is determined by the fixed process of data tag generation and upload and does not change with the optimization of the auditing mechanism. The improved scheme in this paper achieves the best performance in the TPA→CS, CS→TPA, and on-chain interaction phases.

8.4.3 Challenge Phase Overhead Comparison

Table 7 details the differences between the original scheme and the improved scheme in the challenge phase in terms of the number of computational operations and communication data volume.

Table 7: Challenge phase overhead comparison.

Comparison	Original Scheme	Improved Scheme
Computational Operations	Hash Operation: 6 times Pseudorandom Function Call: 3	Hash Operation: 4 times Pseudorandom Function Call: 2
Communication Data Volume	Total: 176 Bytes Effective Data Ratio: 81.82%	Total: 144 Bytes Effective Data Ratio: 100%

9 Conclusion

This paper focuses on the core issue of insufficient tag security in cloud storage data integrity auditing. Targeting the tag forgery vulnerability caused by the lack of dynamic random parameters in the hash function input of the blockchain-based certificateless auditing scheme proposed by Miao et al., a systematic improvement study is conducted. Firstly, by in-depth analyzing the tag generation mechanism of the original scheme, multiple attack paths such as tag forgery, data tampering, illegal insertion, and deletion induced by static hash inputs are clarified. Secondly, a lightweight enhancement scheme is proposed, which incorporates the dynamic random parameter T into the input of the hash function H_3, making tag generation depend on both static attributes and dynamic factors, thereby fundamentally blocking the possibility of attackers precomputing and forging tags. Subsequently, under the Computational Diffie-Hellman (CDH) and Discrete Logarithm (DL) hardness assumptions, the scheme's resistance against Type I, Type II, and Type III adversaries is verified through formal proofs, ensuring core security properties such as tag unforgeability, collusion resistance, and data privacy protection. Finally, experimental performance analysis confirms

that the improved scheme only introduces negligible computational overhead, and the efficiency of the proof generation and verification phases is improved compared with the original scheme, maintaining good practicality.

The core contribution of this paper lies in achieving a significant enhancement of security with minimal modifications. In the blockchain decentralized auditing scenario, the scheme achieves dynamic unforgeability of tags with minimal modifications (only one additional hash operation), while maintaining full compatibility with blockchain smart contracts. This avoids the sharp increase in on-chain computing overhead caused by substantial protocol modifications, and effectively solves the contradiction between "security improvement and efficiency loss" of existing randomization methods in blockchain scenarios. It not only retains the key management advantages of certificateless cryptography and the decentralized transparency characteristics of blockchain from the original scheme but also effectively compensates for the security flaws in the tag generation phase, providing a data integrity auditing solution that balances high security and practicality for multi-cloud storage environments.

Future research can be further expanded in three aspects: first, exploring the lightweight optimization of the scheme in resource-constrained scenarios such as edge computing and IoT terminals to further reduce the local computational overhead of tag generation; second, integrating zero-knowledge proof technology to enhance data privacy protection during the auditing process, achieving the dual goals of verifiable audit results and zero leakage of data content; third, expanding the cross-chain auditing capability of the scheme to adapt to the complex multi-cloud storage architecture with heterogeneous blockchains, and improving the compatibility and scalability of the scheme in large-scale distributed storage environments.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by Engineering University of PAP's Funding for Education and Teaching Program Grant (No. Wjx2025069), Engineering University of PAP's Funding for Basic and Cutting-Edge Innovation Grant (No. Wjy202520) and Engineering University of PAP's The Second Batch of Scientific Research and Innovation Teams. This work is also supported by Stability Program of National Key Laboratory of Security Communication (WD202513).

Author Contributions: Conceptualization: Chao Zhang and Xu An Wang; Methodology: Chao Zhang and Weiwei Jiang; Software: Weidong Zhong; Validation: Ziteng Wang and Miao Tian; Formal analysis: Jianhong Ling; Investigation: Chao Zhang; Resources: Hangjiang Du; Data curation: Chao Zhang; Writing—original draft preparation: Chao Zhang; Writing—review and editing: Chao Zhang; Visualization: Chao Zhang; Supervision: Yunhui Duan; Project administration: Chao Zhang; Funding acquisition: Weidong Zhong. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data supporting the findings of Section 8 are available from the corresponding author, Weiwei Jiang (Email: jww@bupt.edu.cn), upon reasonable request.

Ethics Approval: This study did not involve any human or animal subjects, and therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mell P, Grance T. The NIST definition of cloud computing. 2011 [cited 2026 Jan 1]. Available from: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=909616.
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. A view of cloud computing. Commun ACM. 2010;53(4):50–8. doi:10.1145/1721654.1721672.

3. Hashizume K, Rosado DG, Fernández-Medina E, Fernandez EB. An analysis of security issues for cloud computing. J Internet Serv Appl. 2013;4(1):5. doi:10.1186/1869-0238-4-5.
4. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security; 2007 Oct 29–Nov 2; Alexandria, VA, USA. p. 598–609. doi:10.1145/1315245.1315318.
5. Juels A, Kaliski BS Jr. Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security; 2007 Oct 29–Nov 2; Alexandria, VA, USA. p. 584–97. doi:10.1145/1315245.1315317.
6. Shacham H, Waters B. Compact proofs of retrievability. J Cryptol. 2013;26(3):442–83. doi:10.1007/s00145-012-9129-2.
7. Wang Q, Wang C, Ren K, Lou W, Li J. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst. 2011;22(5):847–59. doi:10.1109/tpds.2010.183.
8. Yuan J, Yu S. Efficient public integrity checking for cloud data sharing with multi-user modification. In: Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications; 2014 Apr 27–May 2; Toronto, ON, Canada. p. 2121–9. doi:10.1109/infocom.2014.6848154.
9. Wang C, Wang Q, Ren K, Cao N, Lou W. Toward secure and dependable storage services in cloud computing. IEEE Trans Serv Comput. 2012;5(2):220–32. doi:10.1109/tsc.2011.24.
10. Ali H, Abidin S, Alam M. Auditing of outsourced data in cloud computing: an overview. In: Proceedings of the 2024 11th International Conference on Computing for Sustainable Global Development (INDIACom); 2024 Feb 28–Mar 1; New Delhi, India. p. 111–7. doi:10.23919/indiacom61295.2024.10498177.
11. Wang C, Chow SSM, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput. 2013;62(2):362–75. doi:10.1109/tc.2011.245.
12. Yoosuf MS, Anitha R. LDuAP: lightweight dual auditing protocol to verify data integrity in cloud storage servers. J Ambient Intell Humaniz Comput. 2022;13(8):3787–805. doi:10.1007/s12652-021-03321-7.
13. Zheng Z, Xie S, Dai HN, Chen X, Wang H. Blockchain challenges and opportunities: a survey. Int J Web Grid Serv. 2018;14(4):352. doi:10.1504/ijwgs.2018.095647.
14. Yue D, Li R, Zhang Y, Tian W, Huang Y. Blockchain-based verification framework for data integrity in edge-cloud storage. J Parallel Distrib Comput. 2020;146(22):1–14. doi:10.1016/j.jpdc.2020.06.007.
15. Huang P, Fan K, Yang H, Zhang K, Li H, Yang Y. A collaborative auditing blockchain for trustworthy data integrity in cloud storage system. IEEE Access. 2020;8:94780–94. doi:10.1109/access.2020.2993606.
16. Miao Y, YingMiao QH, Qiong Huang MX, Meiyan Xiao WS. IPAPA: incentive public auditing scheme against procrastinating auditor. J Internet Technol. 2022;23(7):1505–17. doi:10.53106/160792642022122307006.
17. Liu Z, Feng Y, Ren L, Zheng W. Data integrity audit scheme based on blockchain expansion technology. IEEE Access. 2022;10:55900–7. doi:10.1109/access.2022.3176754.
18. Zhang Q, Zhang Z, Cui J, Zhong H, Li Y, Gu C, et al. Efficient blockchain-based data integrity auditing for multi-copy in decentralized storage. IEEE Trans Parallel Distrib Syst. 2023;34(12):3162–73. doi:10.1109/tpds.2023.3323155.
19. Zhu Y, Wang H, Hu Z, Ahn GJ, Hu H, Yau SS. Dynamic audit services for integrity verification of outsourced storages in clouds. In: Proceedings of the 2011 ACM Symposium on Applied Computing; 2011 Mar 21–24; TaiChung, Taiwan. p. 1550–7. doi:10.1145/1982185.1982514.
20. Yang K, Jia X. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distrib Syst. 2013;24(9):1717–26. doi:10.1109/tpds.2012.278.
21. Zhou C, Wang L, Wang L. Lattice-based provable data possession in the standard model for cloud-based smart grid data management systems. Int J Distrib Sens Netw. 2022;18(4):155013292210929. doi:10.1177/15501329221092940.
22. Yang J, Wang H, Wang J, Tan C, Yu D. Provable data possession of resource-constrained mobile devices in cloud computing. J Netw. 2011;6(7):1033–40. doi:10.4304/jnw.6.7.1033-1040.
23. Li J, Yan H, Zhang Y. Identity-based privacy preserving remote data integrity checking for cloud storage. IEEE Syst J. 2021;15(1):577–85. doi:10.1109/jsyst.2020.2978146.
24. Yuan Y, Zhang J, Xu W, Li Z. Identity-based public data integrity verification scheme in cloud storage system via blockchain. J Supercomput. 2022;78(6):8509–30. doi:10.1007/s11227-021-04193-6.

25. Li J, Wu J, Jiang G, Srikanthan T. Blockchain-based public auditing for big data in cloud storage. Inf Process Manag. 2020;57(6):102382. doi:10.1016/j.ipm.2020.102382.
26. Xu S, Cai X, Zhao Y, Ren Z, Wu L, Zhang H, et al. zkrpChain: privacy-preserving data auditing for consortium blockchains based on zero-knowledge range proofs. In: Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom; 2020 Dec 29–2021 Jan 1); Guangzhou, China. p. 656–63. doi:10.1109/trustcom50675.2020.00092.
27. Chang J, Shao B, Ji Y, Bian G. Efficient identity-based provable multi-copy data possession in multi-cloud storage, revisited. IEEE Commun Lett. 2020;24(12):2723–7. doi:10.1109/lcomm.2020.3013280.
28. Li T, Hu L. Audit as you go: a smart contract-based outsourced data integrity auditing scheme for multiauditor scenarios with one person, one vote. Secur Commun Netw. 2022;2022(3):8783952–13. doi:10.1155/2022/8783952.
29. Tahir M, Sardaraz M, Muhammad S, Saud Khan M. A lightweight authentication and authorization framework for blockchain-enabled IoT network in health-informatics. Sustainability. 2020;12(17):6960. doi:10.3390/su12176960.
30. Miao Y, Miao Y, Miao X. Blockchain-based transparent and certificateless data integrity auditing for cloud storage. Concurr Comput. 2024;36(27):e8285. doi:10.1002/cpe.8285.
31. Nweje U. Blockchain technology for secure data integrity and transparent audit trails in cybersecurity. Int J Res Publ Rev. 2024;5(12):4902–16. doi:10.55248/gengpi.5.1224.0211.
32. Li S, Xu C, Zhang Y, Du Y, Chen K. Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage. IEEE Trans Serv Comput. 2022;16(1):134–46. doi:10.1109/tsc.2022.3144430.
33. Zhu C, Lu Y, Xia N, Li J, Sun Y. A lightweight blockchain-assisted certificateless cloud data integrity auditing scheme without third-party auditor. IEEE Trans Inform Forensic Secur. 2026;21:976–91. doi:10.1109/tifs.2026.3652010.
34. Wang L, Guan Z, Chen Z, Hu M. Enabling integrity and compliance auditing in blockchain-based GDPR-compliant data management. IEEE Internet Things J. 2023;10(23):20955–68. doi:10.1109/jiot.2023.3285211.
35. Shen W, Gai C, Yu J, Su Y. Keyword-based remote data integrity auditing supporting full data dynamics. IEEE Trans Serv Comput. 2024;17(5):2516–29. doi:10.1109/tsc.2023.3339521.
36. Tu Z, Wang X, Du W, Wang Z, Lv M. An improved multi-copy cloud data auditing scheme and its application. J King Saud Univ Comput Inf Sci. 2023;35(3):120–30. doi:10.1016/j.jksuci.2023.01.021.
37. Kumar RP, Bandanadam SR. Block chain-based decentralized public auditing for cloud storage with improved EIGAMAL encryption model. Int J Inf Technol. 2024;16(2):697–711. doi:10.1007/s41870-023-01599-8.
38. Wang L, Hu M, Jia Z, Guan Z, Chen Z. SStore: an efficient and secure provable data auditing platform for cloud. IEEE Trans Inform Forensic Secur. 2024;19:4572–84. doi:10.1109/tifs.2024.3383772.
39. Liu Z, Wang S, Liu Y. Blockchain-based integrity auditing for shared data in cloud storage with file prediction. Comput Netw. 2023;236:110040. doi:10.1016/j.comnet.2023.110040.
40. Zhang Q, Qian S, Cui J, Zhong H, Wang F, He D. Blockchain-based privacy-preserving deduplication and integrity auditing in cloud storage. IEEE Trans Comput. 2025;74(5):1717–29. doi:10.1109/tc.2025.3540670.
41. Zhou H, Shen W, Liu J. Certificate-based multi-copy cloud storage auditing supporting data dynamics. Comput Secur. 2025;148(3):104096. doi:10.1016/j.cose.2024.104096.
42. Miao Y, Gai K, Zhu L. Blockchain-assisted multi-keyword searchable provable data possession for cloud storage. Sci China Inf Sci. 2026;69(3):132101. doi:10.1007/s11432-024-4409-6.
43. Vijayakumar D, Srinivasagan KG, Vivekrabinson K. Enhancing cloud storage security through blockchain-enabled data deduplication and auditing with a fair payment. Peer Peer Netw Appl. 2025;18(3):147. doi:10.1007/s12083-025-01970-5.
44. Xu C, Feng L, Jing Z, Huang F, Yu Y. PATD: privacy-preserving auditing and transparent deduplication in UAV cloud storage. Comput Secur. 2026;166(3):104905. doi:10.1016/j.cose.2026.104905.