Hindawi
Security and Communication Networks
Volume 2023, Article ID 5477738, 21 pages
https://doi.org/10.1155/2023/5477738

Research Article

A Low-Overhead Auditing Protocol for Dynamic Cloud Storage Based on Algebra

Fudong Ding,[1,2,3] **Libing Wu**,[1,2,3] **Zhuangzhuang Zhang**,[1,2,3] **Xianfeng Wu**,[4] **Chao Ma**,[1,3] **and Qin Liu**[1,3]

[1]*School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China*
[2]*Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Guangzhou 510000, China*
[3]*Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan 430072, China*
[4]*Institute for Interdisciplinary Research, Jianghan University, Wuhan 430056, China*

Correspondence should be addressed to Libing Wu; wu@whu.edu.cn and Zhuangzhuang Zhang; zhzhuangzhuang@whu.edu.cn

Received 10 December 2022; Revised 28 March 2023; Accepted 13 May 2023; Published 3 June 2023

Academic Editor: Wenxiu Ding

Copyright © 2023 Fudong Ding et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the widespread adoption of cloud storage, ensuring the integrity of outsourced data has become increasingly important. Various cloud storage auditing protocols based on public key cryptography have been proposed. However, all of them require complex cryptographic operations and incur significant storage and communication costs. To address the issues of significant storage overhead for data tags, high computational complexity of cryptographic algorithms, and limited efficiency of dynamic data algorithms in signature algorithm-based cloud storage outsourcing data integrity verification protocols, we propose a dynamic auditing protocol called AB-DPDP, which is based on algebra. Our protocol reduces the computational complexity of tag generation by utilizing basic algebraic operations instead of the traditional cryptographic method used in most current auditing protocols. To reduce storage overhead and protect private data, our protocol stores only tags, allowing for data to be restored through these tags, as opposed to storing both tags and data on the cloud server. To accommodate for more frequent and efficient data dynamics, we propose the dynamic index skip table data structure. Furthermore, the security of our proposed protocol is thoroughly proven based on the security definition of secure cloud storage. Finally, through theoretical analysis and experimental evaluation, we demonstrate the advantages of our scheme in terms of data privacy, storage overhead, communication overhead, computation overhead, and data dynamic efficiency.

1. Introduction

With the advancement of cloud computing technology, cloud storage services have become increasingly popular among both enterprise and individual users [1]. The reasons for storing data in cloud servers vary; while enterprises may choose to outsource their data to reduce the cost of purchasing hard drives, personal users prefer cloud storage for the convenience of accessing their data from multiple devices, as well as to save local storage space, such as on their mobile phones where audio and video data can consume a lot of space. However, data outsourcing also presents numerous security challenges. Although cloud service providers (CSPs) offer secure hosting environments and robust security mechanisms, outsourced data still face threats such as cyberattacks, hardware failures, and software vulnerabilities [2, 3]. Furthermore, in an effort to reclaim storage space, CSPs may discard or compress rarely accessed data, and in some cases, even deceive users by claiming that the outsourced data are intact following a data breach event, in order to maintain their reputation. As a result, it becomes critical for cloud storage users to have the ability to verify the integrity of their outsourced data.

Current research efforts have suggested several key metrics for secure cloud storage: (1) Efficiency: the entire auditing protocol should have a low time complexity. (2) Support for third-party auditing: a trusted third party can reduce the computational overhead of the user by performing the auditing tasks on their platform. Additionally, in case of data loss, the audit results from a trusted third party are more impartial and objective compared to those from both CSPs and users. (3) Data dynamics: since users often update the data stored in the cloud, a secure cloud storage protocol should be able to support data dynamics. In conclusion, a secure cloud storage protocol that is computationally efficient, can support dynamic data, and allows for third-party public auditing is needed.

Typically, the data owner must generate tags for the file blocks before outsourcing the data for subsequent verification of their integrity. Both the data blocks and their tags are uploaded to the cloud service provider (CSP). The outsourced data consist of both data and tags. Although we can adjust the size of tags by changing the size of the data blocks, the communication and storage overheads of the protocol are still greatly affected by them. For example, in the RSA-based scheme [4], if the bit length of each data block and tag is set to 1024, the size of the outsourced data will be twice the size of the data block. Furthermore, as shown in Table 1, the majority of existing protocols utilize RSA or BLS signature algorithms [14] for tag generation, resulting in high computational complexity, and none of these protocols can adequately safeguard users' private data from being accessed by CSPs. Additionally, to resist replacement attack, the block index is embedded in the tag of each data block. In general, the dynamic operation of data changes the block index, requiring the tags of all subsequent data blocks to be recalculated during integrity verification. This results in a significant computational overhead. To address this issue, a mapping must be established between the tag index and block index, where the tag index is used for tag calculation and the block index represents the storage location of the block. This dynamic approach avoids tag recalculation and is therefore highly efficient. However, current solutions for index mapping have several drawbacks: (1) linked list [15], where index conversions are done by traversing backward from the head of the linked list; (2) index array [16], where all subsequent entries in the index array need to be moved when a data block is inserted; and (3) dynamic operation array [17], where the complexity of the index conversion algorithm is high when data dynamics are too frequent. The disadvantages mentioned previously pose significant challenges to deploying cloud storage auditing protocols in real-world scenarios.

To address these challenges, it is essential to introduce a more efficient auditing scheme for cloud storage. An ideal solution would be to store only tags, not data blocks, in the CSP to minimize communication and storage overhead while protecting users' data privacy. To improve tag computation, integrity proof computation, and proof verification efficiency, we aim to reduce the time-consuming exponentiation and bilinear pair operations in existing schemes. Additionally, to support frequent data dynamic operations, we must reduce the time complexity of index conversion data structures. The Goldreich–Goldwasser–Halevi (GGH) cryptosystem [18] is a lattice-based public key encryption algorithm, where both the private and public keys are metrics. This system is capable of encrypting data matrices using the private key and recovering the original data using the public key. We propose an algebraic equation that draws inspiration from the GGH cryptosystem. By replacing the public and private keys in GGH encryption with big integers, we can easily verify the integrity of an input big integer and recover it through the encrypted data. However, verifying one data block at a time is not efficient enough for verification. Therefore, we improve the proposed algebraic equation to support the verification of multiple data blocks by accumulating the encrypted data blocks. To improve data dynamic efficiency, we must reduce the time complexity of dynamic operations of data structures. We achieve this by improving the skip table, a data structure that supports binary search, to dynamically adjust the update strategy based on the amount of data to achieve data dynamic operations in logarithmic time complexity.

We have developed a novel auditing protocol for cloud storage that uses basic algebraic operations for tag calculation, data integrity verification, and data restoration. Our approach minimizes storage overhead and enhances users' data privacy by adopting a tag-only storage approach in the CSP. To improve data dynamic efficiency, we propose a dynamic index skip table (DIST) that supports index transformation and reduces the time complexity of the data dynamic algorithm. We start by proposing an algebraic equation and improving it to support aggregated verification of multiple data blocks. We then introduce the architectures of cloud storage private and public auditing protocols and discuss the design goals of a secure auditing protocol. We construct the AB-DPDP private auditing protocol based on the proposed algebraic equation and enhance it to support public auditing by introducing a TPA and a PKG. Finally, we propose a new technique to support data dynamics such as insertion, deletion, or modification of data blocks. Our data structure dynamically adjusts the update cost of data block insertion based on the size of the data volume and enables index conversion in logarithmic time complexity.

The main contributions of this paper are as follows:

(i) We make the first attempt to propose a basic algebraic equation and, based on it, a novel secure cloud storage protocol, AB-DPDP (algebra-based provable data possession). Compared to existing schemes, this protocol involves only basic algebraic operations, reducing the computational complexity for tag computation and integrity verification. Additionally, it requires only the storage of tags on the cloud server, significantly reducing storage and communication overhead and enhancing privacy protection.

(ii) We propose a new data structure, the dynamic index skip table, to enhance the AB-DPDP protocol, resulting in a cloud storage auditing scheme that supports efficient data dynamic operations. This improvement enhances the efficiency of data block search, insertion, deletion, and updating compared to existing data dynamic schemes.

Table 1: Limitations of various typical integrity verification protocols.

Protocols	Public auditing	Data dynamic operations	Cryptographic algorithm	Corrupt data recovery	Privacy protection
Ateniese et al. [4]	✗	✗	RSA	✗	✗
Shacham and Waters [5] and Liu et al. [6]	✗	✗	PRF/BLS	✓	✗
Ateniese et al. [7]	✗	Limited	PRF, HASH	✗	✗
Wang et al. [8]	✓	✓ (MHT)	RSA	✗	✗
Shen et al. [9]	✓	✓ (Linked list)	BLS	✗	✗
Rabaninejad et al. [10]	✓	✓ (MHT)	Proxy resignature	✗	✗
Zhang et al. [11]	✓	✗	Blockchain, BLS	✗	✗
Ramaiah and Kumari [12]	✓	✓ (Linked list)	Homomorphic encryption	✗	✗
Hahn et al. [13]	✓	✓ (index conversion)	Homomorphic encryption	✗	✗

(iii) We formalize the system framework of private and public auditing protocols and the security model of AB-DPDP. We theoretically analyze the protocol in terms of its correctness, privacy, and efficiency and demonstrate its ability to prevent forgery, replay, and replacement attacks. Furthermore, we conduct extensive experiments to evaluate the performance of the protocol and show that it significantly reduces storage and communication costs during data outsourcing and greatly improves the efficiency of dynamic data handling.

2. Related Work

In this section, we show works related to cloud storage integrity auditing protocols, focusing on public auditability and data dynamics.

2.1. No Support for Data Dynamics. Ateniese et al. [4] proposed a provable data possession protocol (PDP) where a public verifier can disclose the abnormal data incidents of CSPs with no user data. PDP utilizes an aggregable homomorphic linear authenticator (HLA) based on RSA to calculate tags of outsourced data. Users are able to verify a linear combination of individual data blocks through the aggregated HLAs. Using a sampling strategy, the verifier can quickly check the integrity of the outsourced data with a certain probability. Juels and Kaliski [19] proposed proofs of retrievability (POR). POR not only supports data integrity verification but also allows recovering corrupted data by erasure codes. However, POR introduces a special block called sentinel that is randomly embedded in the data, limiting the operation of data updates. To address these issues, Shacham and Waters [5] proposed two improved POR schemes that overcome the limitations of POR and provide security proof. The first private verification scheme is built on pseudo-random function (PRF), and the second public verification scheme is constructed based on the BLS signature algorithm [14]. Liu et al. [6] proposed a scheme based on regenerated codes where users are able to audit a random subset of the outsourced data. However, none of the above schemes supports the dynamic operation of data.

2.2. Support for Data Dynamics. To support dynamic operations, Ateniese et al. [7] designed an extensible PDP, which supports limited data dynamic operations. Erway et al. [20] studied the dynamic PDP scheme and refined the proposed scheme in a subsequent work [21]. They designed a dynamic provable data possession (DPDP) scheme, which supports data updates by introducing a rank-based authenticated skip table. Esiner et al. [22] improved this work by using variable lists to support variable block size updates. Wang et al. [8] proposed a public auditing protocol for dynamic cloud storage that combines HLA with Merkle hash tree (MHT) to support data dynamics. To reduce the burden of frequent auditing for users, a trusted third-party auditor (TPA) is introduced to support public auditing, and TPA is also extended by many subsequent schemes. However, in this scheme, once the data are updated, the MHT needs to be rebuilt. To support fine-grained updates, Liu et al. [23] proposed a dynamic public auditing scheme based on Merkle trees which are used to support variable-sized block updates. Zhu et al. [24] proposed a dynamic auditing scheme based on fragment structure and index hash table. Unfortunately, in this scheme, the time complexity of data structure updating is high when data are updated.

Other research lines aiming to improve the efficiency of data dynamics are batch updates [25], index transformations [26], and dynamic hash tables [9, 27]. The repetitive computations associated with data dynamics can be avoided by batch updates, but the update delay is unacceptable in applications requiring real-time performance. Index transformation methods provide a way to convert the block index to a tag index, while dynamic-hash-table-based methods aim to decouple indexes from tags. However, the computational overhead of data dynamics for all these schemes increases linearly with the number of challenged blocks.

2.3. Limitations of Recent Research. Recent research has focused on enhancing protocol security and improving other features such as protocol scalability on top of supporting public auditable basis. Wang et al. [28] proposed an auditing scheme for outsourced database. The scheme uses Embedded Merkle B tree and Bloom filter to authenticate and verify the search results with correctness and completeness properties. However, the scheme does not consider data dynamics. Some studies focus on resilience to key leakage [29], public auditing of shared data [10], and blockchain-based auditing [11, 30, 31]. In order to strengthen the protection of users' private data against untrusted TPA, random masking [32], homomorphic hash function [33], and homomorphic encryption [12] were adopted. Although these solutions enhanced security and scalability in terms of multiple users, they have not yet achieved efficient data dynamics.

To reduce the cost of certificate management, some protocols [34–36] adopted identity-based encryption and signature schemes. Erway et al. [21] improved a dynamic PDP scheme that supports data dynamics, but the data owner is also involved in the authentication computation, and Guo et al. [25] improved this scheme by introducing TPA. Hahn et al. [13] proposed a more efficient cloud auditing scheme by introducing precomputation in the TPA validation phase shifting part of the validation computation from CSP to TPA, and the two computations are performed independently and concurrently to improve the efficiency of validation, but the scheme does not reduce the actual communication and computation overhead. Yang et al. [17] proposed a dynamic cloud auditing scheme based on a dynamic operation array. This scheme uses a dynamic operation array to implement index transformation, which is more efficient when dynamic operations are less frequent. However, as dynamic operations tend to be more frequent, the computational complexity of its index transformation increases linearly. Javadpour et al. [37] proposed a more efficient task scheduling method, which optimized energy consumption by 12% and power consumption by 20%. Sangaiah et al. [38] proposed a feature selection method used for classifying the malicious and legitimate activities in cloud computing environments with higher accuracy, which enhanced security for cloud storage environments.

To more clearly demonstrate the limitations of various integrity verification protocols, we have summarized the comparison results in Table 1.

3. Protocol Design

The cloud storage integrity auditing protocol proposed in this paper is based on the algebraic identity in Theorem 1 and supports tag generation, data integrity verification, data restoration, and dynamic data handling. In this section, we will focus on Theorem 1 and design a private auditing protocol based on Corollary 2 and the secure cloud storage model. Finally, the private auditing protocol is improved to support public auditing and dynamic data handling.

3.1. Theorem 1. To reduce the storage overhead of cloud storage auditing protocols, an ideal solution is to perform integrity verification only through data tags, without the help of raw data blocks. To achieve this, we introduce Theorem 1, which serves as the basis of the proposed integrity auditing protocol. The theorem does not require complex cryptographic operations, only simple algebraic operations, to detect changes in input large integers.

Theorem 1. *let $\sigma = wd + r$, where d is the input big integer, w is the private key, and r is the disturbance variable.*

Introduce $e \in \mathbb{Z}_N{}^*$, let $P = e\sigma$, if $w > 2er$, then $P = \mathrm{round}(P/w) \cdot w + er$, $d = \mathrm{round}(P/w)/e$.

Proof

$$P = e\sigma = e(wd + r) = ewd + er, \tag{1}$$

for $w > 2er$, then

$$\mathrm{round}\left(\frac{P}{w}\right) = \mathrm{round}\left(ed + \frac{er}{w}\right) = ed. \tag{2}$$

According to equations (1) and (2),

$$\begin{aligned}\mathrm{round}\left(\frac{P}{w}\right) \cdot w + er &= ewd + er\\ &= e(wd + r) = P,\end{aligned} \tag{3}$$

$$\frac{\mathrm{round}(P/w)}{e} = d.$$

The proof is completed. □

According to Theorem 1, we can derive

Corollary 2. *For l linearly related large integers*

$$\sigma_i = wd_i + r_i, 1 \leqslant i \leqslant l, \tag{4}$$

let

$$\Sigma = \sum_{i=1}^{l} e_i\sigma_i, e_i \in \mathbb{Z}_N{}^*, \tag{5}$$

if $w > 2\sum_{i=1}^{l} e_i r_i$, then

$$\Sigma = \mathrm{round}\left(\frac{\Sigma}{w}\right) \cdot w + \sum_{i=1}^{l} e_i r_i. \tag{6}$$

Proof

$$\begin{aligned}\Sigma = \sum_{i=1}^{l} e_i\sigma_i &= \sum_{i=1}^{l} e_i(wd_i + r_i)\\ &= w\sum_{i=1}^{l} e_i d_i + \sum_{i=1}^{l} e_i r_i,\end{aligned} \tag{7}$$

$$\begin{aligned}\mathrm{round}\left(\frac{\Sigma}{w}\right) &= \mathrm{round}\left(\sum_{i=1}^{l} e_i d_i + \frac{\sum_{i=1}^{l} e_i r_i}{w}\right)\\ &= \sum_{i=1}^{l} e_i d_i + \mathrm{round}\left(\frac{\sum_{i=1}^{l} e_i r_i}{w}\right),\end{aligned} \tag{8}$$

for $w > 2\sum_{i=1}^{l} e_i r_i$, then

$$\mathrm{round}\left(\frac{\sum_{i=1}^{l} e_i r_i}{w}\right) = 0. \tag{9}$$

According to equations (8) and (9),

$$\mathrm{round}\left(\frac{\Sigma}{w}\right) = \sum_{i=1}^{l} e_i d_i. \tag{10}$$

According to equations (7) and (10),

$$\Sigma = \mathrm{round}\left(\frac{\Sigma}{w}\right) \cdot w + \sum_{i=1}^{l} e_i r_i. \tag{11}$$

The proof is completed. □

3.2. Secure Cloud Storage Model. In cloud storage environments, the integrity of user data in CSPs is exposed to various threats, and to maintain their reputation and interests, CSPs may conceal data corruption events from users. Therefore, data owners must be able to check the integrity of their data in a timely manner. To provide more impartial proof to CSPs and users, audit tasks are often delegated to TPAs. Introducing a trusted private key generator (PKG) to distribute keys based on the identity of users can help reduce the burden of certificate management. Therefore, after the user outsources their data to the CSP, TPA is responsible for auditing the data integrity in the CSP according to a certain frequency and returning the results to the user.

Based on these requirements, we present the classic models for AB-DPDP private and public auditing protocols. The private auditing protocol involves two entities: user and CSP. The user calculates data tags, uploads data tags to the CSP, challenges the CSP, and verifies the proofs returned by the CSP. The CSP receives and stores data tags, accepts challenges, and generates proofs. The public auditing protocol involves four entities: PKG, user, CSP, and TPA. The PKG is responsible for user key generation and management. The user calculates data tags and uploads data and tags to the CSP. The CSP receives and stores user data and tags, accepts challenges from the TPA, and generates proofs. The TPA receives the user's audit request, challenges the CSP, verifies the proofs returned by the CSP, and returns the verification result to the user.

As depicted in Figures 1 and 2, the AB-DPDP private and public auditing protocols are encompass seven stages, namely setup, key generation, data outsourcing, challenge initiation, proof calculation, proof verification, and data recovery, collectively represented as AB-DPDP = (Setup, KeyGen, Outsource, Challenge, Proof, Verify, Recovery).

(i) $\mathrm{Setup}(1^\lambda) \Rightarrow (\mathrm{PK}, \mathrm{SK})$: This algorithm is run by PKG, which takes security parameters λ as input and generates public key PK and private key SK.

(ii) $\mathrm{KeyGen}(\mathrm{SK}, \mathrm{UID}) \Rightarrow \mathrm{SK}_{\mathrm{UID}}$: In public auditing protocol, this algorithm is run by PKG, which utilizes the private key SK and user identity UID to generate the user private key $\mathrm{SK}_{\mathrm{UID}}$. In private auditing protocol, this algorithm is run by user, which chooses a private key SK.

(iii) $\mathrm{Outsource}(d^*, \mathrm{SK}_{\mathrm{UID}}, \mathrm{PK}) \Rightarrow \sigma$: This algorithm is run by user, which takes user file d^*, public key, and user private key as input and output the outsourced data and then uploads it to CSP.

(iv) $\mathrm{Challenge}(1^\lambda) \Rightarrow c^*$: This algorithm is run by user or TPA, which generates a challenge sequence based on the input security parameters and sends the sequence to CSP.

(v) $\mathrm{Proof}(\sigma, c^*, \mathrm{PK}) \Rightarrow P$: This algorithm is run by CSP, which generates integrity proof based on the challenge sequence from TPA, outsourced data, and public key and then sends the proof to TPA.

(vi) $\mathrm{Verify}(c^*, P, \mathrm{PK}, \mathrm{SK}_{\mathrm{UID}}) \Rightarrow v$: This algorithm is run by user or TPA, which verifies the integrity proof with the challenge sequence and outputs the verification result (complete/corrupt) which will be sent to user.

(vii) $\mathrm{Recovery}(i, \mathrm{SK}) \Rightarrow t_i \Rightarrow d_i$: This algorithm is run by user, which restores the original data blocks based on the tags.

A secure integrity auditing protocol should aim to meet the following six design goals:

(1) Correctness: the protocol should ensure that CSPs with no data loss will always pass the integrity audit

(2) Efficiency: the protocol should have low time and spatial complexity to minimize the overhead of storage, communication, and computing

(3) Antispoofing: the protocol should be able to detect data corruption even if CSPs attempt to deceive users, with a high probability

(4) Privacy protection: the protocol should ensure that neither the CSP nor the TPA can obtain the raw data from the outsourced data

(5) Data accessibility: users should be able to obtain their raw data at any time

(6) Data dynamics: the protocol should allow users to modify, add, or delete outsourced data and still be able to audit the updated data

3.3. AB-DPDP Private Auditing Protocol.

The most important aspect of designing an auditing protocol is generating an integrity proof that can be verified. In most current schemes, users must attach additional verification information (tags) to each data block in order to calculate the proof. However, this means that the CSP must store both the data and the tags, leading to heavy storage and communication overhead. To alleviate this issue, we propose a secure cloud storage private auditing protocol called AB-DPDP. This protocol is based on Theorem 1 and its corollary and meets the requirements of secure cloud storage. Unlike other protocols, AB-DPDP only stores tags in the CSP and uses simple algebraic operations to calculate both the tags and the integrity proofs.

As shown in Figure 1, the AB-DPDP private auditing protocol involves only two participants: the user and the CSP. The protocol consists of six algorithms: KeyGen, Outsource, Challenge, Proof, Verify, and Recovery. The main steps of the protocol are as follows:

(1) $\mathrm{KeyGen}(1^\lambda) \Rightarrow SK$: Based on the security parameters λ, the user selects a random large integer $s_k \in \mathbb{Z}_p{}^*$ and a hash function $H_1\colon \{0,1\}^* \longrightarrow \mathbb{Z}_q$, where $p > 2q$. The private key is $\mathrm{SK} = (s_k, H_1)$.

(2) $\mathrm{Outsource}(d^*, SK) \Rightarrow \sigma$: The user first divides the file into blocks $d_i \in \mathbb{Z}_N{}^*, 0 \leqslant i \leqslant n$ according to a fixed size, where n is the number of data blocks. Then, the tags are calculated for each data block $t_i = s_k d_i + r_i$, where $r_i = H_1(i), 0 \leqslant i \leqslant n$; finally, the user uploads the tag collection $\sigma = \{t_i\}^*$ to the CSP and deletes local data and tags.

(3) $\mathrm{Challenge}(1^\lambda) \Rightarrow c^*$: The user selects l data blocks by random sampling to initiate a challenge and generate a corresponding random number for each sampling block $e_i < \lambda, 1 \leqslant i \leqslant l$, where $2\sum_{j=1}^{l} e_{i_j} r_{i_j} < w$ and form a challenge sequence $c^* = (i_l, i_2, \ldots, i_l; e_1, e_2, \ldots, e_l)$ to challenge CSP.

(4) $\mathrm{Proof}(\sigma, c^*) \Rightarrow P$: The CSP retrieves the corresponding tags according to the challenge sequence, calculates the integrity proof $P = \sum_{i=1}^{l} e_i t_i$, and then returns the proof to user for verification.

(5) $\mathrm{Verify}(P, SK) \Rightarrow v$: The user verifies the integrity proof P received by judging whether the equation $round(P/s_k) \cdot s_k + \sum_{i=1}^{l} e_i r_i = P$ holds or not. If it holds, then output $v = true$, that is, the data are complete. Otherwise, output $v = false$.

(6) $\mathrm{Recovery}(i, SK) \Rightarrow t_i \Rightarrow d_i$: The user enters the block index i, and the CSP returns the corresponding tag t_i.

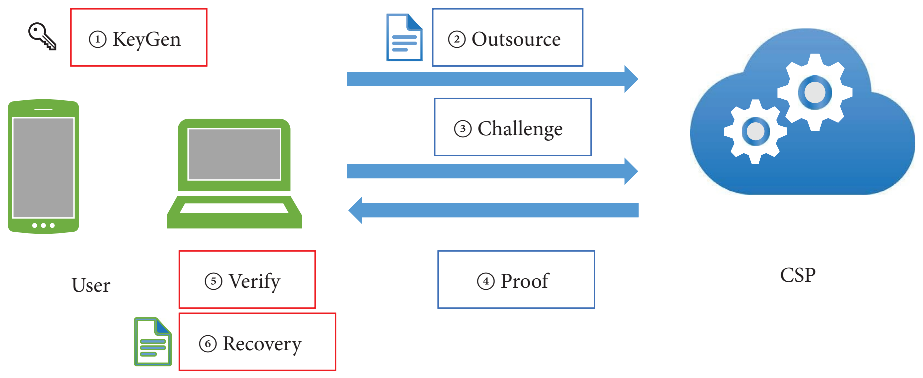


Figure 1: AB-DPDP private auditing protocol.

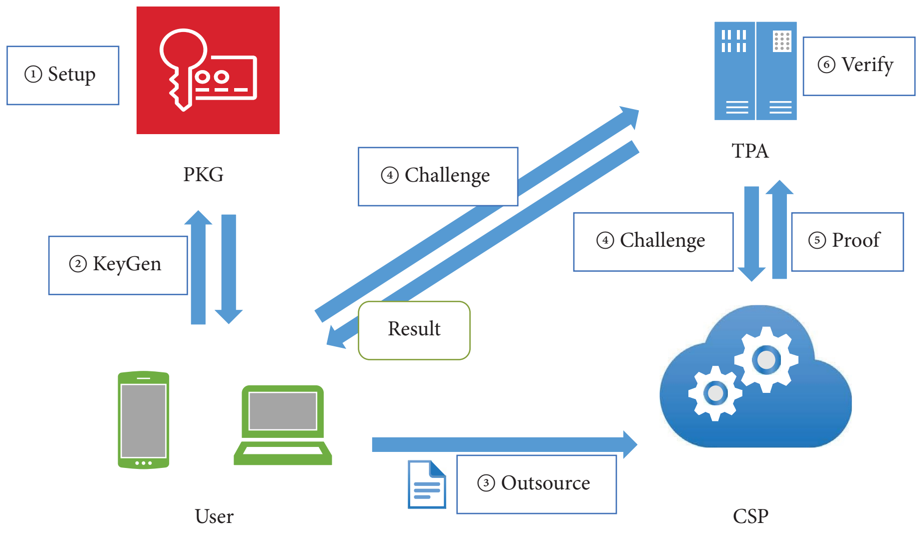


Figure 2: AB-DPDP public auditing protocol.

The user then restores the raw data block d_i through the tag and private key SK, $d_i = \text{round}(t_i/s_k)$.

3.4. AB-DPDP Public Auditing Protocol. The participants of the AB-DPDP private auditing protocol are limited to the user and CSP, which results in the user being required to participate in the calculation of the challenge initiation and integrity proof verification. This imposes a heavy computational overhead on the user. Additionally, since the audit results are issued by the user, the CSP may question the accuracy of the results, even if the data are corrupt. To reduce the computational overhead on the user and provide more impartial audit results, we improved the AB-DPDP private auditing protocol to the public auditing protocol in Figure 2 by introducing the TPA and PKG.

The AB-DPDP public auditing protocol consists of four entities: the user, PKG, TPA, and CSP and includes seven algorithms: setup, key generation, data outsourcing, challenge initiation, proof generation, proof verification, and data recovery. In the public auditing protocol, the setup and key generation are executed by the PKG, and the calculation of challenge initiation and proof verification is transferred to the TPA, which provides fair and credible audit results. With the introduction of PKG and TPA, the protocol must also accommodate the conversion from a single-user scenario to a multiuser scenario. The main algorithms of the public auditing protocol are as follows:

(1) $\text{Setup}(1^\lambda) \Rightarrow (\text{PK}, \text{SK})$: PKG selects three security hash functions $H_1, H_2\colon \{0,1\}^* \longrightarrow \mathbb{Z}_p, H_3\colon \{0,1\}^* \longrightarrow \mathbb{Z}_q$, based on the security parameters λ, where $p > 2q$. Let private key $\text{SK} = H_1$, public key $\text{PK} = (H_2, H_3)$, and send the public key to user and TPA.

(2) $\text{KeyGen}(\text{SK}, \text{uid}) \Rightarrow \text{SK}_{\text{uid}}$: PKG runs this algorithm to generate user private key. The user sends identity information uid to the PKG, which uses the private key hash function to generate the user's private key $\text{SK}_{\text{uid}} = H_1(\text{uid})$, and then returns it to the user.

(3) $\text{Outsource}(d^*, \text{SK}_{\text{uid}}, \text{PK}) \Rightarrow \sigma$: The user first divides the file into blocks $d_i \in \mathbb{Z}_N{}^*, 0 \leqslant i \leqslant n$ according to a fixed size, where n is the number of data blocks. Then, the user generates a random large integer R_{uid}, and the tags are calculated for each data block $t_i = p_k \cdot (H_1(\text{SK}_{\text{uid}}|i) \cdot d_i) + r_i$, where $p_k = H_2(R_{uid}), r_i = H_3(i), 0 \leqslant i \leqslant n$. Finally, the user

uploads the tag collection $\sigma = \{t_i\}^*$ and R_{uid} to the CSP and deletes local data and tags.

(4) Challenge $(1^\lambda) \Rightarrow c^*$: Sampling the data blocks and challenging the CSP. This algorithm is consistent with the algorithm in private auditing protocol.

(5) Proof $(\sigma, c^*) \Rightarrow P$: The integrity proof is calculated according to the challenge sequence and the corresponding tags. This algorithm is consistent with the algorithm in private auditing protocol.

(6) Verify $(P, \text{PK}, R_{uid}) \Rightarrow v$: The user verifies the integrity proof P received by judging whether the equation $\text{round}(P/p_k) \cdot p_k + \sum_{i=1}^{l} e_i r_i = P$, where $p_k = H_2\ (R_{uid}), r_i = H_1(i)$, holds or not. If it holds, then output $v = true$, that is, the data are complete. Otherwise, output $v = false$.

(7) Recovery $(i, \text{SK}) \Rightarrow t_i \Rightarrow d_i$: The user enters the block index i, and the CSP returns the corresponding tag t_i. The user then restores the raw data block d_i through the tag and private key SK, $d_i = (\text{round}(t_i/p_k))/(H_1(SK_{uid}|i))$.

3.5. Data Dynamics. Both the AB-DPDP private and public auditing protocols support the verification of static data, but in a real cloud storage environment, dynamic data operations are frequent. Since dynamic operations affect the mapping relationship between block index and tag index, a data structure is needed to record this mapping to support dynamic auditing. Index conversion schemes such as arrays, linked lists, hash tables, and dynamic operation arrays have been proposed, but they have high data structure update complexity when the amount of data and the frequency of dynamic operations are high. In this paper, we introduce the dynamic index skip table (DIST) to dynamically adjust the data structure update strategy based on the number of data blocks, resulting in more efficient data dynamics.

As shown in Figure 3(a), a node in the DIST consists of a block index, a tag index, and multilayer pointers. The block index is the actual storage index of the tag in the CSP, while the tag index is used to calculate the tag before data outsourcing. The nodes are linked by multilayer pointers, and the lowest layer must have two pointers pointing to the previous and next nodes, while other layers generate a pointer to the next node with a certain probability (the upper layer has a smaller probability) to reduce the complexity of node search time. If a node has only one level, the block index of that node is set to −1 and will not be updated during dynamic data operations to reduce the time complexity of data dynamics. The generation probability P of the upper-layer pointers is not fixed but dynamically decreases or increases as the number of nodes increases or decreases. For example, when the number of nodes is 1000, 2-layer pointers are generated with a probability of 1/4, and 3-layer pointers with a probability of 1/16. When the number of nodes is 2000, 2-layer nodes are generated with a probability of 1/8, and 3-layer pointers with a probability of 1/64. By dynamically adjusting the upper-layer pointer generation probability, the number of nodes with upper-layer pointers does not increase linearly with the increase in data volume. To improve node search efficiency, if the absolute difference between the found node's block index and the target node's block index is less than $1/P$, the bottom bidirectional pointer is used to find it forward or sequentially.

The AB-DPDP protocol uses the dynamic index skip table (DIST) to support dynamic data operations, including updates, additions, and deletions of data blocks. Algorithm 1 outlines how the AB-DPDP protocol uses DIST to achieve this, and Algorithm 2 outlines the update strategy for the DIST data structure during dynamic data operations.

Figure 3 shows the process of updating DIST when the data are dynamic. Figure 3(a) shows the initial state of DIST, and Figure 3(b) shows the state after the insertion of node 3, with the red node being the newly inserted node and its tag index being equal to the number of data blocks plus one. Since there is no upper-level pointer, the block index of the newly inserted node is set to −1. Figure 3(c) shows the state after the deletion of node 4. It can be seen that during insertion and deletion operations, only the nodes that come after the current node and have upper-level pointers will update their block indexes. If we were to query the node with block index 5 in Figure 3(c), according to Algorithm 1, we would first search the top layer, but not find a near node. We would then search the next layer, find node 6, and see that the distance between node 6 and the target node is less than 2. We would only need to search one node forward from node 6 to find the target node 5.

The integrity auditing process has the following changes with the introduction of DIST:

(i) The tag index used by the Outsource algorithm to calculate the tag must be consistent with the index used in update and insert operations of Algorithm 1

(ii) Before the Verify algorithm calculates $\sum_{i=1}^{l} e_i r_i, r_i = H_3(i)$, it should convert the sampled block index collection $c^* = \{i_1, \ldots, i_l\}$ into the corresponding tag index collection $c^{*'} = \{i_1', \ldots, i_l'\}$ through the DIST $\text{dist} = \{i_j : i_j'\}^*, j \in (1, n)$

4. Security Analysis

Cloud storage data integrity is frequently exposed to multiple security threats, among which attacks against integrity verification protocols are also common. Therefore, the proposed protocols must be designed to be resistant to these attacks. In this section, we formalize the security model of the protocol into two security definitions and then the correctness of the protocols, and their resistance to forgery attacks, replacement attacks, and replay attacks are analyzed, respectively, in accordance with security Definition 3. Furthermore, the protection of user data by the proposed protocols from the perspective of data security is analyzed based on security Definition 4.

4.1. Security Model. Based on the design goals of secure cloud storage, the security model of the cloud storage integrity verification protocol can be formalized using the following two security definitions.

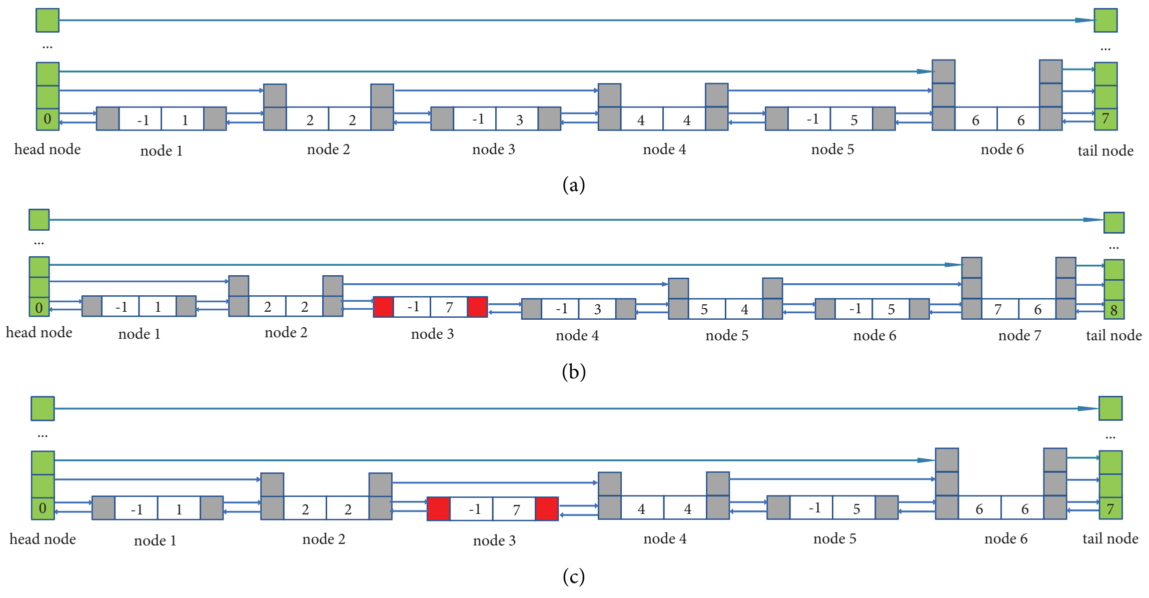


Figure 3: Dynamic operations of DIST: (a) initial DIST, (b) insert node 3, and (c) delete node 4.

```
Input: index $i$ and data dynamic operation
$op$(insert/ delete/modify).
Output: data dynamic results.
(1) Initial next_idx = $n + 1$, collection of available indexes idx_set = $\varnothing$, initial dynamic index skip table DIST;
(2) if op == modify then
(3)     The user sends instruction $(\text{modify}, i)$ to TPA and CSP, then CSP returns the tag $t_i$;
(4)     The user uses the recovery algorithm to restore the original data block $d_i$, modifies the data block, and generates a new tag. The calculation rules of the new tag are as follows:
(5)     if idx_set $\neq \varnothing$ then
(6)         choose one element $i'$ from idx_set;
(7)     else
(8)         $i'$ = next_idx, next_idx = next_idx + 1;
(9)     end if
(10)    $t_i' = p_k \cdot (H_1(SK_{uid}|i') \cdot d_i) + r_i$, where $r_i = H_3(i')$;
(11)    The user sends the new tag to CSP to replace the old tag and sends the new tag index $i'$ to TPA;
(12)    TPA take $(\text{modify}, i, i')$ as the input of Algorithm 2 to update the DIST;
(13) end if
(14) if op == delete then
(15)    The user sends instruction $(\text{delete}, i)$ to TPA and CSP;
(16)    CSP deletes the tag corresponding to the specified block index $i$;
(17)    TPA take $(\text{delete}, i)$ as the input of Algorithm 2 to update the DIST;
(18) end if
(19) if op == insert then
(20)    The user calculates the tag $t_i$ for the inserting data block;
(21)    if idx_set $\neq \varnothing$ then
(22)        choose one element $i'$ from idx_set;
(23)    else
(24)        $i'$ = next_idx, next_idx = next_idx + 1;
(25)    end if
(26)    $t_i = p_k \cdot (H_1(SK_{uid}|i) \cdot d_i) + r_i$, where $r_i = H_3(i')$;
(27)    The user sends instruction $(\text{insert}, i, i')$ to TPA and sends tag and block index to the CSP;
(28)    TPA takes $(\text{insert}, i, i')$ as the input of Algorithm 2 to update the DIST;
(29) end if
```

Algorithm 1: AB-DPDP data dynamic algorithm.

```
Input: Operation $op$ (search/insert/ delete/modify), target block index $i_b$, tag index $i_t$. DIST head node phead, and tail node ptail.
The second-layer pointer threshold $PN$ and the upper-layer pointer maximum generation probability $P_{\max}$.
Output: Target node/DIST update results.
(1) Initial node = phead;
(2) if $op$ == search then
(3)   Let next be the top pointer;
(4)   while node.next.block_idx < $i_b$ do
(5)     $node = node.next$
(6)     if |node.block_idx − $i_b$| < = $1/\min(PN/n+1, P_{\max})$ then
(7)       near_node = node;
(8)       end while;
(9)     end if
(10)  end while
(11)  if near_node == NULL then
(12)    Let $next$ be the next layer pointer, repeat 4–10;
(13)  end if
(14)  Search |near_node.block_idx − $i_b$| nodes forward or backward by the bidirectional pointer to locate the target node target_node;
(15)  return target_node
(16) end if
(17) if $op$ == $insert$ then
(18)   $op(search(i_b))$ = > $target\_node$, insert the new node $inode(i_b, i_t)$ before $target\_node$;
(19)   Generate a k-layer pointer with the probability of $P = \min(PN/n+1, P_{\max})^k$ and link the new node to DIST. let $i_b = -1$ if $k_{\max}$ == 1;
(20)   Update the block index of node with second-layer pointer backward by increasing the block index by 1;
(21) end if
(22) if $op$ == delete then
(23)   $op$(search($i_b$)) = > target_node, delete(target_node). Relink the pointer of each layer. idx_set.push(target_node.tag_idx);
(24)   Update the block index of node with second-layer pointer backward by decreasing the block index by 1;
(25) end if
(26) if $op$ == modify then
(27)   $op$(search($i_b$)) = > target_node, and modify target_node.tag_idx = $i_t$;
(28) end if
```

ALGORITHM 2: DIST update algorithm.

Definition 3. Antispoofing. If the probability of adversary A1 defeating challenger C1 in the security game G1 is negligible under the security parameter λ, i.e., $P(G1) = negl(\lambda)$, then the cloud storage integrity verification protocol is considered to have antispoofing.

4.1.1. Game G1. As shown in Figure 4, Game G1 involves an adversary A1 and a challenger C1.

(i) Setup: C1 runs the setup algorithm to generate the master private key and public key and sends the public key to A1.

(ii) Query phase: A1 is allowed to make two types of queries to C1 multiple times. (1) KeyGen Query: A1 queries the private key of identity ID from C1. C1 runs the KeyGen algorithm to generate the user's private key based on ID and returns it to A1. (2) Outsource query: A1 can query the encrypted file block corresponding to a random file F. C1 runs the outsource algorithm to generate the encrypted file block based on the input file F from A1 and returns it to A1.

(iii) Challenge: C1 runs the challenge algorithm to generate a challenge sequence and sends it to A1.

(iv) Forge: A1 forges a proof P and returns it to C1.

(v) Output: If A1's forged proof passes C1's verification, A1 is considered to have won game G1.

Definition 4. Privacy protection. If the probability of adversary A2 defeating challenger C2 in the security game G2 is negligible under the security parameter λ, then the cloud storage integrity verification protocol is considered to have the ability to protect user privacy.

4.1.2. Game G2. As shown in Figure 5, Game G2 involves an adversary A2 and a challenger C2.

(i) Setup: C2 runs the setup algorithm to generate the master private key and public key and sends the public key to A2.

(ii) Query Phase: A2 is allowed to make two types of queries to C2 multiple times, similar to Game G1.

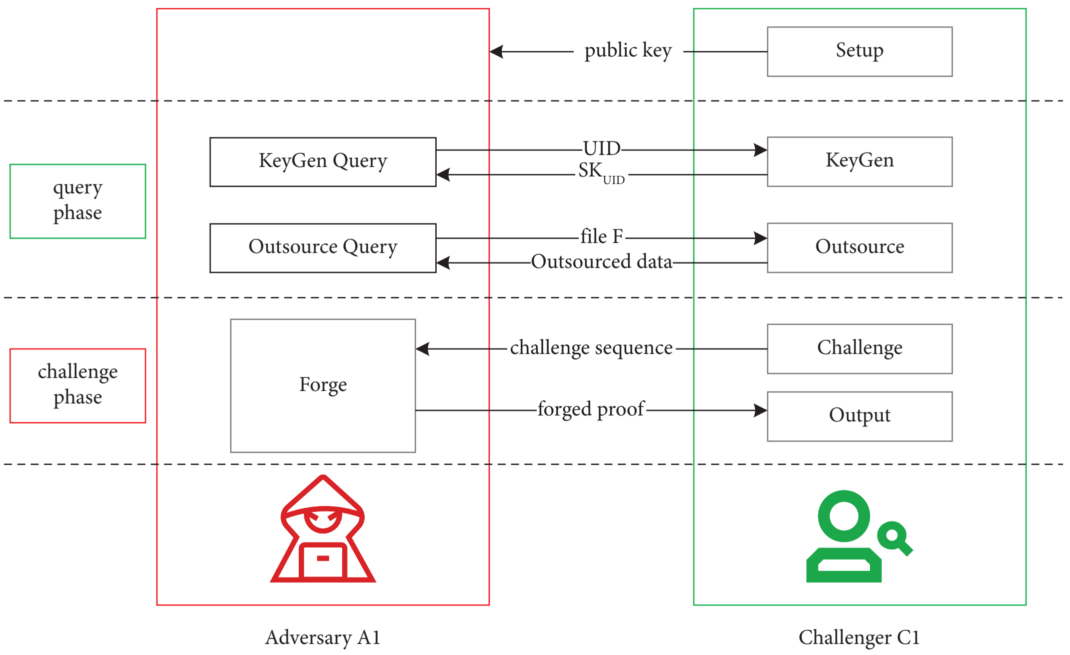


FIGURE 4: Security game G1: antispoofing.

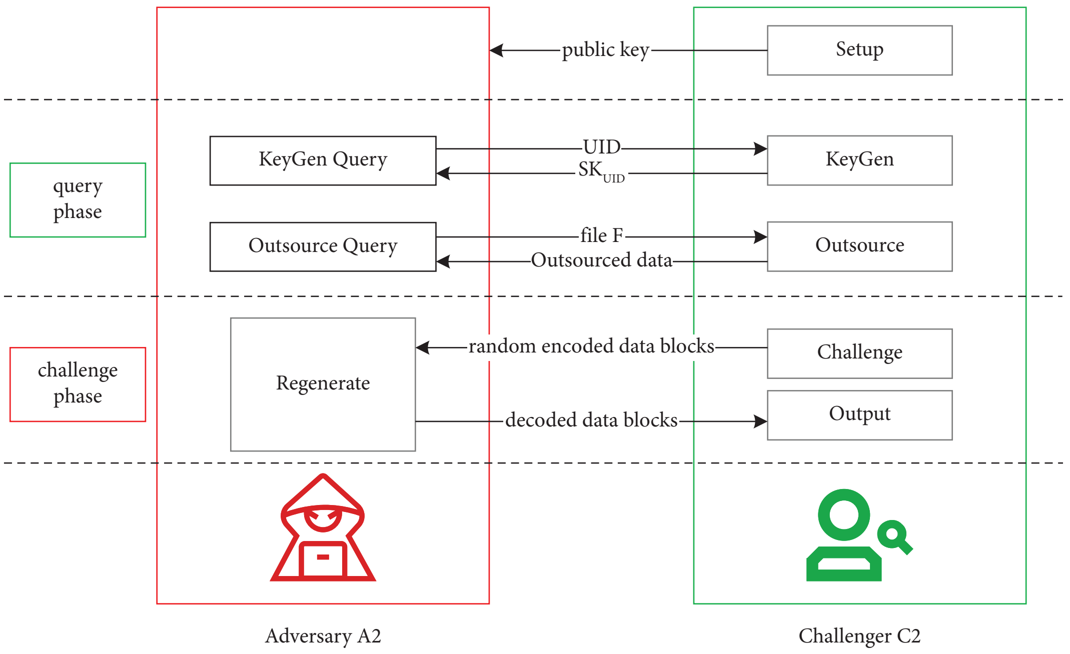


FIGURE 5: Security game G2: privacy protection.

(iii) Challenge: C2 randomly selects data blocks and generates encoded data blocks using the Outsource algorithm. These data blocks are then sent to A2.

(iv) Regenerate: A2 generates the original data blocks based on the encoded data blocks and sends them to C2.

(v) Output: If the data blocks generated by A2 match the data blocks selected by C2, A2 is considered to have won Game G2.

4.2. Correctness. Correctness refers to the fact that a cloud service provider (CSP) with complete data always passes an integrity verification when both the user and the trusted third-party auditor (TPA) act as trusted participants. The AB-DPDP protocol determines data integrity by verifying that the equation $\text{round}(P/w) \cdot w + \sum_{i=1}^{l} e_i r_i = P$ holds, and according to Theorem 1 and its corollary, it always makes the equation hold if the tag in the CSP is complete.

4.3. Forgery Attacks. According to security Definition 3, the security model for forgery attacks can be defined as Game A:

(1) The TPA challenges the CSP after sampling data blocks
(2) The CSP uses forged block index tags to calculate the integrity proof P_{forge}
(3) The TPA uses Verify algorithm to verify the integrity proof P_{forge}

If P_{forge} passes the Verify algorithm, the CSP will be considered the winner of Game A.

In order to achieve a secure integrity verification protocol, the probability of the CSP winning game A should be negligible, i.e., $P(A) = negl(\lambda)$.

From Theorem 1, when $\sigma = wd + r, P = e\sigma$, if $w > 2er$, then

$$P = \text{round}\left(\frac{P}{w}\right) \cdot w + er. \tag{12}$$

Does equation (12) still hold when σ is tampered with? Suppose σ is tampered with as $\sigma = wd + r + diff$, then

$$P = e\sigma = e(wd + r + \text{diff}), \tag{13}$$

$$\text{round}\left(\frac{P}{w}\right) \cdot w + er = \text{ewd} + ew \cdot \text{round}\left(\frac{\text{diff}}{w}\right) + er. \tag{14}$$

From equations (13) and (14), if equation (12) holds, it is only necessary to ensure that $\text{round}(\text{diff}/w) = \text{diff}/w$, i.e., diff is an integer multiple of w.

From the above analysis, it is clear that to achieve a forgery attack, the CSP must know w, and w is secured by the security parameter λ. Therefore, the probability of the CSP forging a tag that can pass integrity verification is negligible.

4.4. Replacement Attacks. The difference between the security model of the replacement attack and the forgery attack lies in step (2), where in a forgery attack, the CSP uses a forged tag, while in a replacement attack, the CSP tries to use a tag from another location to compute the integrity proof. If the replacement attack succeeds, the CSP only needs to save a copy of the correct tag to pass the integrity verification.

To resist the replacement attack, the AB-DPDP protocol inserts index information into the tag. Tag $t_i = p_k \cdot (H_1(SK_{uid}|i) \cdot d_i) + r_i, r_i = H_3(i)$. Suppose the challenge block index is i and the CSP performs a replacement attack using a tag with index j. Then,

$$P_j = et_j = ep_k \cdot (H_1(SK_{uid}|j) \cdot d_i) + eH_3(j). \tag{15}$$

The integrity proof verification formula is

$$P_j = \text{round}\left(\frac{P_j}{p_k}\right) \cdot p_k + eH_3(i). \tag{16}$$

From equations (15) and (16), if equation (16) holds, it must satisfy $H_3(j) = H_3(i)$. Obviously, the replacement attack cannot succeed.

4.5. Replay Attacks. The difference between the security model of the replay attack and the other two attacks lies in that it does not attempt to construct a tag that can pass verification. Instead, after a successful verification, the CSP saves the calculated integrity proof and directly uses the saved proof for future verifications. If the replay attack succeeds, the CSP can still pass verification even after deleting all the sampled blocks.

In order to resist replay attacks, the AB-DPDP protocol generates random numbers during each verification and ensures the uniqueness of each generated proof through the use of these random numbers.

Assuming that the TPA initiates two rounds of verification and the verified data blocks are the same, the CSP performs a replay attack in the second round by directly returning the first proof. The first proof is

$$P_i = e_a t_i = e_a p_k \cdot (H_1(SK_{uid}|i) \cdot d_i) + e_a H_3(i). \tag{17}$$

If the second verification directly returns P_i, the TPA will verify it according to the following equation:

$$P_i = \text{round}\left(\frac{P_i}{p_k}\right) \cdot p_k + e_b H_3(i). \tag{18}$$

According to equation (17), if equation (18) holds, $e_a = e_b$ must be satisfied. Since both e_a and e_b are large integers generated randomly under the security parameter, the probability of a successful replay attack is negligible.

4.6. Data Security. According to security Definition 4, data security refers to the fact that the probability of TPA and CSP obtaining the user's original data without the user's involvement should be negligible. In the AB-DPDP protocol, TPA samples l data blocks for integrity verification and obtains l tags, i.e., $t_i = p_k \cdot (H_1(SK_{uid}|i) \cdot d_i) + H_3(i), 0 \leq i \leq l$. Although TPA knows the public key p_k, it is almost impossible for TPA to recover the original data block as the private key SK_{uid} and the original data block d_i are unknown.

5. Performance Analysis and Experiment

In this section, we aim to demonstrate the superiority of the AB-DPDP protocol through both theoretical analysis and experiments. We will focus on evaluating its performance in terms of storage overhead, computing overhead, communication overhead, and data dynamic efficiency. Additionally, we will compare the proposed protocol with existing schemes to further highlight its advantages.

5.1. Performance Analysis. The performance of the AB-DPDP protocol is mainly evaluated by the accuracy of its verification results, the computational complexity, the communication overhead between the various entities involved in the protocol, and the storage overhead incurred by TPA and CSP.

5.1.1. Sampling Algorithm. In order to reduce the heavy time overhead associated with overall-data auditing for large amounts of user data, the AB-DPDP protocol adopts a block-sampling auditing strategy similar to that in the existing scheme [29]. However, sampling auditing does not guarantee the accuracy of auditing results. For example, even if 990 data blocks are sampled from a total of 1000 data blocks, it is still possible to pass the verification if 10 blocks are corrupt. To ensure a high probability of detecting data corruption, our sampling algorithm must be designed carefully. The probability of successful auditing can be expressed using the following formula:

$$\Pr = 1 - \frac{n-t}{n} \cdot \frac{n-t-1}{n-1} \cdots \frac{n-l+1-t}{n-l+1}, \tag{19}$$

where n is the total number of data blocks, t is the number of corrupt data blocks, l is the number of sampling data blocks, and Pr is the probability of detecting the corrupt data.

As illustrated in Figure 6, the probability of successful auditing can be calculated using the formula (19) where n is the number of data blocks and c is the number of sample blocks included in a challenge. As can be seen from the figure, when the data corruption rate is 1/100, by sampling 460 blocks out of 1,000,000 data blocks, it is possible to detect the corrupted block with a probability of 99%. If the data corruption rate is 1/1000, then 4600 data blocks need to be sampled to achieve a 99% error detection rate. To improve the fine-grained and higher probability verification of the AB-DPDP protocol, either the number of sample blocks can be increased or the size of each data block can be enlarged to include more data within the same number of blocks.

5.1.2. Overall Performance. In order to evaluate the overall performance of the proposed AB-DPDP protocol, we compare it with three other schemes: Scheme 1 (RSA-based PDP protocol) [4], Scheme 2 (Pseudo-random function (PRF) based POR protocol), and Scheme 3 (BLS-based POR protocol) [5]. The parameters for comparison are the safety parameter (m), the total number of raw data blocks (m), and the number of sampling data blocks in a challenge (L). The results of this analysis are presented in Tables 2 and 3.

(i) Storage overhead: The proposed protocol reduces the storage overhead compared to other schemes as it only stores tags in the CSP, saving the storage of raw data. However, the introduction of DIST to TPA increases the storage overhead of the data structure compared to the other three schemes, but it is still low compared to the size of raw data. Therefore, the proposed scheme can significantly reduce the storage overhead of the integrity auditing protocol.

(ii) Communication overhead: The communication overhead of the proposed protocol is influenced by the amount of data that needs to be transferred over the network during verification. As both the proposed protocol and the other three schemes adopt sampling verification, the communication overhead is mainly affected by the data uploaded by the user to the CSP. Our protocol, therefore, saves both storage and communication overhead compared to other schemes.

(iii) Computational complexity: The computational complexity of a protocol refers to the time it takes to perform various operations within the protocol. In terms of computational complexity, our proposed protocol has the following advantages. (1) Our protocol requires only three algebraic operations and two hash operations to calculate the tag of each data block, while schemes 1 and 3 require multiple exponentiation operations, leading to higher computational complexity. Scheme 2 requires one exponentiation operation and multiple algebraic operations, resulting in a computational complexity that is comparable to that of our proposed protocol. (2) The challenge sequences of these four schemes are all generated by sampling, so the time complexity of challenge initiation is proportional to the number of data blocks and is the same for all schemes. (3) The time complexity of proof calculation is similar to tag calculation, but the proof calculation is only performed for L sampling data blocks, while tag calculation is performed for all data blocks. (4) Scheme 1 requires three hash operations and one exponentiation operation for proof verification, while scheme 3 requires one bilinear pair operation, resulting in a very high computational complexity. In comparison, our proposed scheme only requires multiple algebraic operations, which requires fewer hash operations than scheme 2, making it the scheme with the lowest computational overhead in proof verification.

5.1.3. Data Dynamics. The performance of data dynamics refers to the performance of data structures that are introduced to support data dynamics, including the time complexity of operations such as insertion, deletion, modification, and search.

In this analysis, we compare the time complexity of various dynamic operations for linked lists, arrays, dynamic operation arrays from previous schemes [15–17], and the DIST introduced in our proposed scheme. Table 4 summarizes the time complexity of data dynamic operations in these four data structures. Assuming that n represents the number of original data blocks and d represents the number of data blocks resulting from data dynamic operations, we assume that the number of new data blocks is proportional to the frequency of data dynamic operations. As shown in the table, the time complexity of all operations on the linked list in the scheme [15] and insertion and deletion operations on the index array in the scheme [16] is proportional to the current total number of data blocks. The time complexity of search and update operations for the index array is O (1). Although the dynamic operation array in [17] only records dynamic operations, its time complexity is proportional to the frequency of dynamic data operations, rather than the

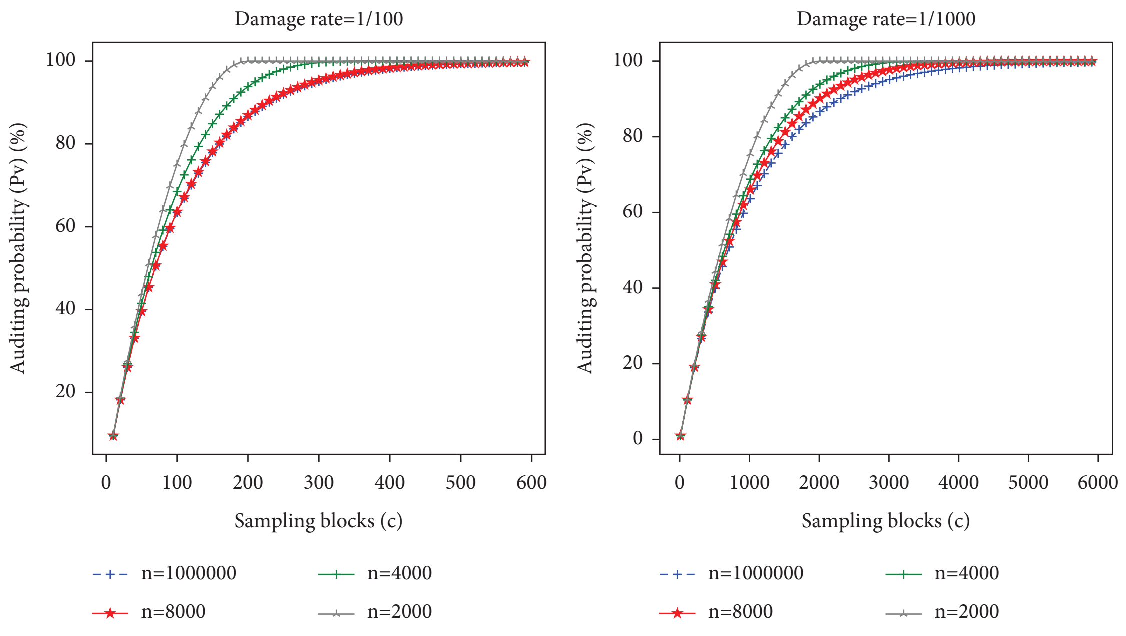


Figure 6: Probability of successful sampling auditing.

total number of data blocks. In contrast, our proposed DIST has an advantage in that it can complete all data dynamic operations under logarithmic complexity even when the data dynamic operations are frequent. Therefore, our proposed protocol is suitable for dynamic integrity verification scenarios in which the total amount of data is large and dynamic operations are frequent.

5.2. Experiment. In this section, we evaluate the performance of the AB-DPDP protocol through several experiments. We implemented the AB-DPDP protocol using C and deployed it on an Ubuntu 16.04 virtual machine equipped with a 4-core 3.3 GHz CPU and 8 GB RAM. The tests are conducted for storage overhead, communication overhead, and computational overhead. To prove the superiority of our protocol, we also conducted comparative experiments on Scheme 1 (RSA-based PDP protocol) [4], Scheme 2 (PRF-based POR protocol), and Scheme 3 (BLS-based POR protocol) [5]. These three protocols serve as the basis for many existing cloud storage integrity auditing protocols. The test dataset consists of random files with sizes ranging from 128 KB to 1 GB, which were generated using the Linux file generation tool. The security parameter, data block size, and maximum number of sampling data blocks were set to 128, 4 KB, and 460, respectively. Meanwhile, the safety parameters of the AB-DPDP protocol's disturbance vector were set to 64. Considering the security requirements of the RSA algorithm, the security parameter of Scheme 1 was set to 1024. At the same time, in order to verify the performance of the proposed DIST data structure, we compared it with arrays and linked lists in terms of data insertion and deletion.

5.2.1. Storage Overhead. The lower the storage overhead, the lower the additional cost for users to use our protocol. In order to save storage overhead, the AB-DPDP protocol adopts a strategy of storing only tags. As depicted in Figure 7(a), the storage overhead of Schemes 1 and 3 is substantial, whereas the storage overhead of Scheme 2 and the proposed AB-DPDP protocol are relatively similar. Our scheme only stores tags in CSP, which results in a lower storage overhead compared to the other three schemes, and the savings in storage overhead increase linearly with the growth of data size.

5.2.2. Communication Overhead. Lower communication overhead means less time for users to upload files and perform integrity verification. As the integrity verification is performed through sampling challenge and the integrity proof is constant, the storage overhead remains the most decisive factor in determining the communication overhead. As shown in Figure 7(b), the communication overhead only increases by the amount of inquiry information and data for integrity proof compared to storage overhead.

5.2.3. Computational Overhead. The computational overhead focuses on the time required to complete an integrity audit, which is mainly comprised of four stages: tag calculation, challenge initiation (sampling), proof calculation, and proof verification. The lower the computational overhead, the shorter the protocol runtime. Computational overhead has a significant impact on the speed of data outsourcing and the timely acquisition of data status by users. In order to reduce the computational overhead of the

Table 2: Computation overhead comparison of integrity auditing protocol.

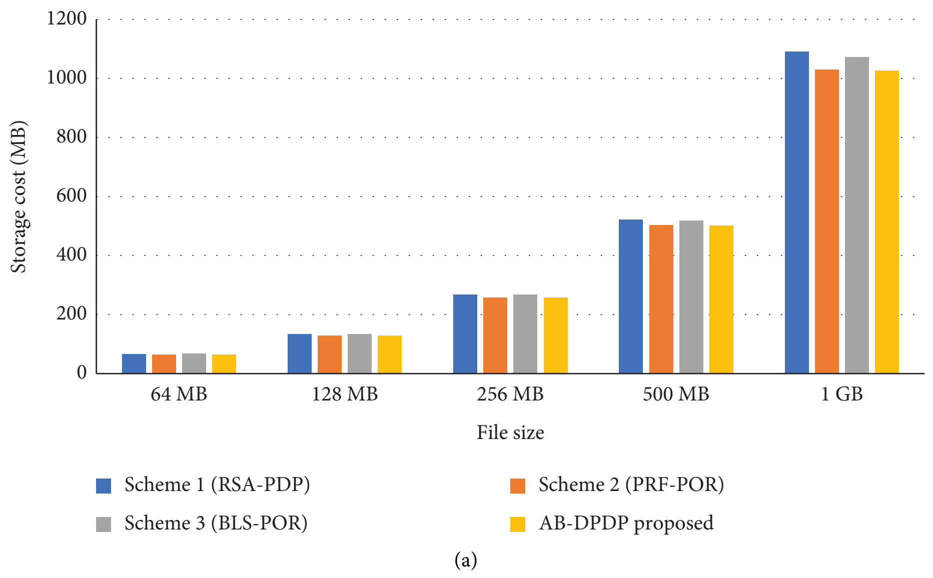
Protocols	Computation overhead			
	Tag calculation	Challenge initiation	Proof calculation	Proof verification
Scheme 1 (RSA-PDP)	$m(2Exp + 1Hash + 1Multi)$	$O(m)$	$LHash + (L + 2)Exp + LAdd + 1Multi$	$3Hash + 1Exp + 1Div$
Scheme 2 (PRF-POR)	$m(1Hash + sMult + +1Add)$	$O(m)$	$(s + L)Mult + LAdd$	$LHash + (s + 1)LMult + 2LAdd$
Scheme 3 (BLS-POR)	$m(1Hash + (s + 1)Mult + (s + 1)Exp)$	$O(m)$	$s(L + 1)Multi + sLAdd + sExp$	$1Pair + LHash + (2L + 1)Mult + 2LExp$
AB-DPDP	$m(2Hash + 2Multi + 1Add)$	$O(m)$	$LMulti + LAdd$	$(L + 1)Mutli + LAdd + 1Div$

Table 3: Storage and communication overhead comparison of integrity auditing protocol.

Protocols	Storage overhead	Communication overhead
Scheme 1 (RSA-PDP)	$2m\lambda$	$2m\lambda + L\lambda + \lambda$
Scheme 2 (PRF-POR)	$2m\lambda$	$2m\lambda + L\lambda + \lambda$
Scheme 3 (BLS-POR)	$2m\lambda$	$2m\lambda + L\lambda + \lambda$
AB-DPDP	$m\lambda + n\log(n)$	$m\lambda + L\lambda + \lambda + n\log(n)$

Table 4: Time complexity of data dynamic operations for different data structures.

Scheme (data structure)	Operation			
	Insertion	Deletion	Update	Search
Scheme [15] (linked list)	$O(n+d)$	$O(n+d)$	$O(n+d)$	$O(n+d)$
Scheme [16] (array)	$O(n+d)$	$O(n+d)$	$O(1)$	$O(1)$
Scheme [17] (dynamic operations array)	$O(d)$	$O(d)$	$O(d)$	$O(d)$
AB-DPDP (DIST)	$O(\lg(n+d))$	$O(\lg(n+d))$	$O(\lg(n+d))$	$O(\lg(n+d))$



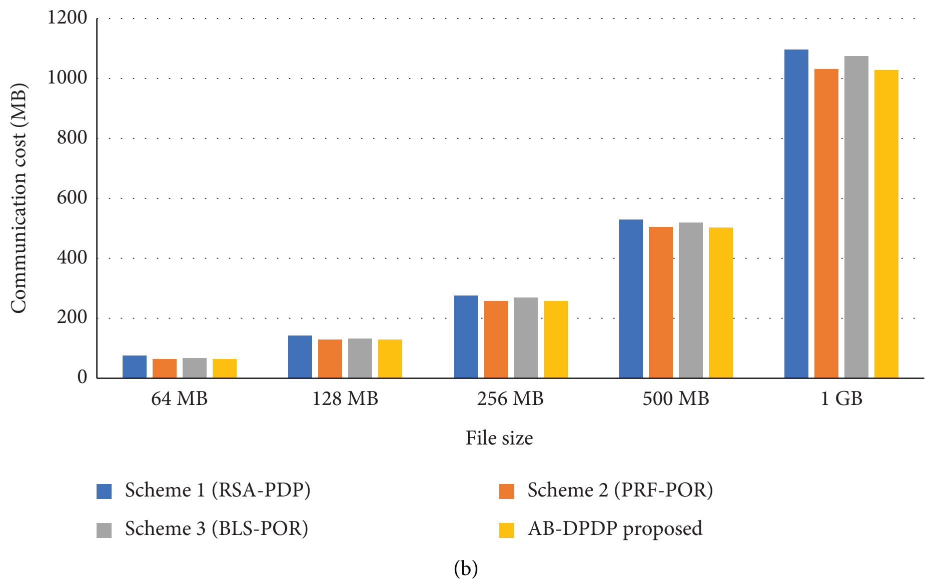
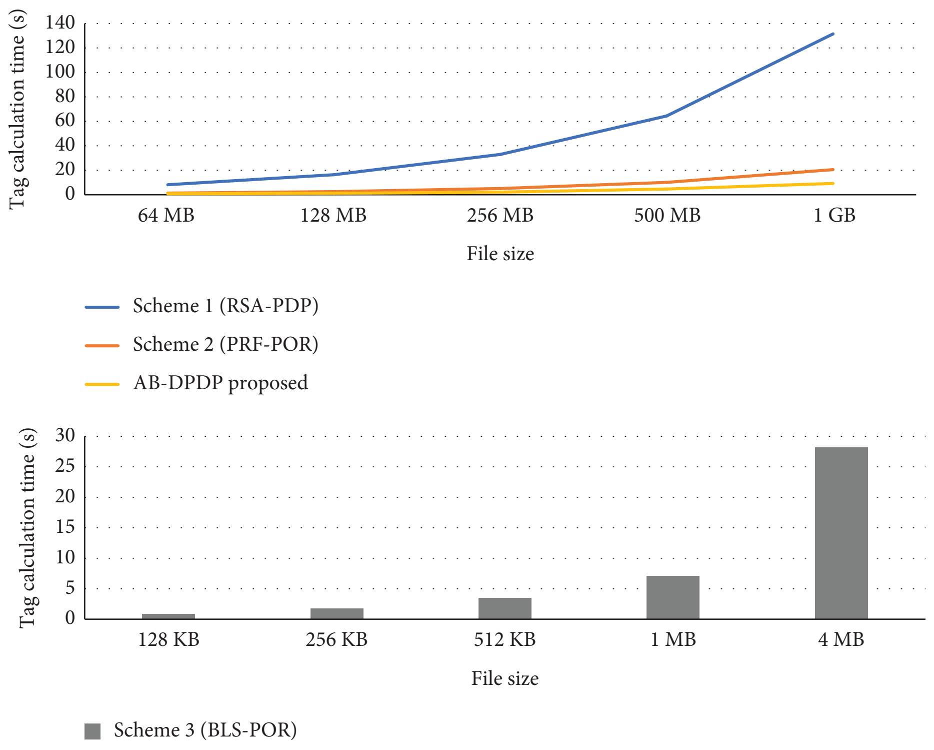
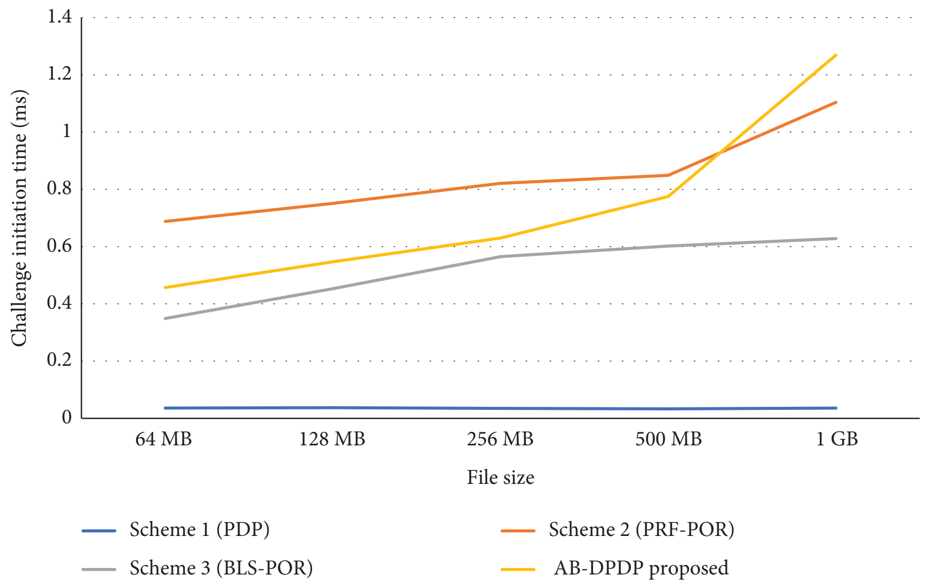


Figure 7: Storage and communication of four schemes: (a) storage overhead and (b) communication overhead.

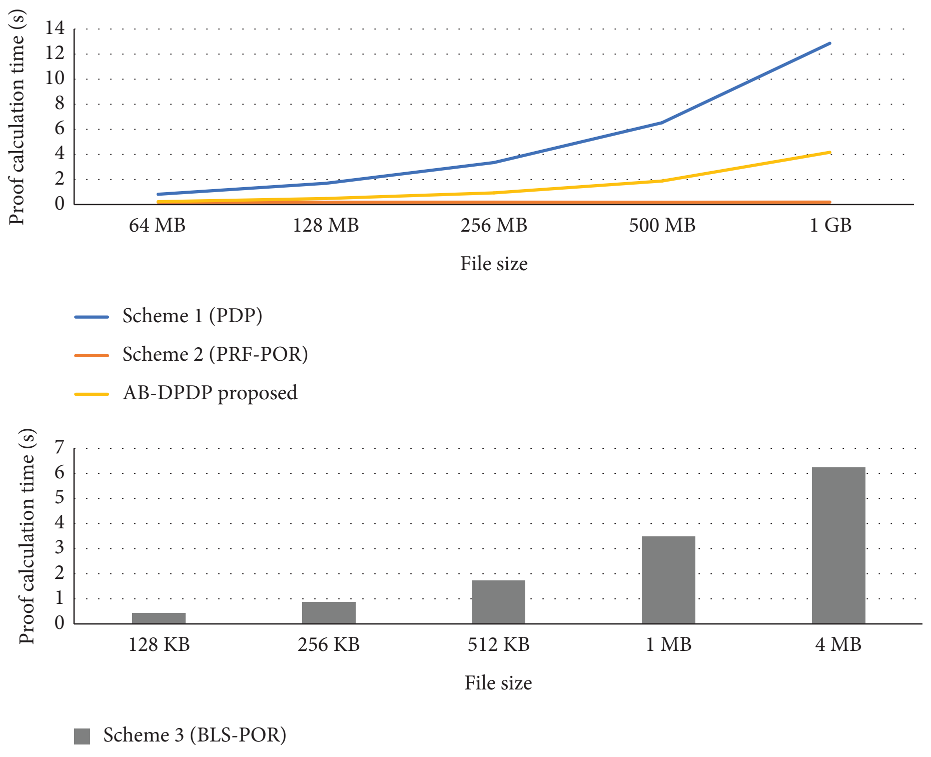


(a)

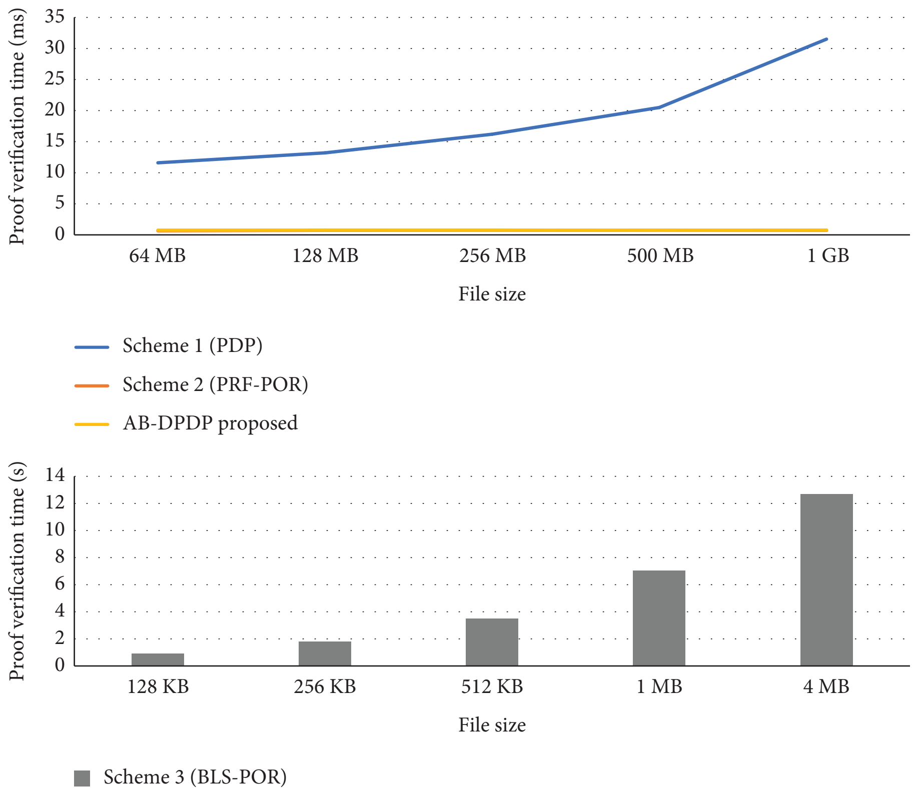


(b)

Figure 8: Continued.



(c)



(d)

Figure 8: Time cost of four algorithms: (a) time consumption of tag calculation, (b) time consumption of challenge initiation, (c) time consumption of proof calculation, and (d) time consumption of proof verification.

Table 5: Time cost of dynamic operations for the data structures: (a) insertion and deletion time cost and (b) update and search time cost.

Operation	Time consumption of 100 insertions (ms)					Time consumption of 100 deletions (ms)				
Number of data blocks ($\times 10^6$)	1	2	3	4	5	1	2	3	4	5
(a)										
Array	740	1504	2211	3032	3722	681	1373	2094	2746	3423
Linked list	723	1485	2141	2877	3703	673	1274	2033	2731	3378
DIST	2.235	3.713	4.996	6.473	8.147	2.147	3.564	5.121	6.387	8.023
Operation	Time consumption of 100 updates (ms)					Time consumption of 100 searches (ms)				
Number of data blocks ($\times 10^6$)	1	2	3	4	5	1	2	3	4	5
(b)										
Linked list	651	1257	2097	2724	3416	591	1256	1941	2597	3313
DIST	2.251	3.658	5.142	6.653	8.312	1.775	2.482	3.328	4.083	5.552

protocol, the AB-DPDP protocol adopts a tag generation scheme based on basic algebraic operations, eliminating complex exponentiation operations and bilinear pair operations. The calculation overheads of the AB-DPDP protocol are as follows:

(i) Tag Calculation. Figure 8(a) demonstrates that compared to the other three schemes, the proposed AB-DPDP protocol has the shortest tag calculation time. Scheme 1 takes longer due to the usage of the RSA signature algorithm, while Scheme 3 takes the longest due to its usage of the BLS signature algorithm. Scheme 2, which only involves hash operations, has a similar calculation time as the proposed protocol. As the file size increases, the difference in calculation time between the four schemes also becomes larger. When the data size reaches 1 GB, the tag calculation time of the AB-DPDP protocol is reduced by nearly 80% compared to the signature algorithm-based protocols.

(ii) Challenge Initiation. Figure 8(b) indicates that all four schemes have similar consumption time for challenge initiation, as they all generate challenge information through sampling. This consumption time is less than 1 ms for all schemes. However, as the number of data blocks increases, the sampling time also increases. When the number of blocks becomes too massive, exceeding the limit of 460 in this study, it may result in a longer sampling time for higher auditing accuracy.

(iii) Proof Calculation. As shown in Figure 8(c), the trend of time consumption for proof calculation is similar to that of tag generation. Scheme 1 and Scheme 3 consume more time in proof generation, especially Scheme 3, which takes significantly more time than the other schemes. The proposed scheme shows significant improvement compared to Scheme 1 and Scheme 3, with only a slight increase in time consumption compared to Scheme 2. When the data size reaches 1 GB, the proof calculation time of the AB-DPDP protocol is reduced by nearly 70% compared to the signature algorithm-based protocols.

(iv) Proof Verification. As shown in Figure 8(d), Scheme 1, Scheme 2, and the proposed scheme have a short verification time, all taking less than 100 ms. Scheme 3 has a longer verification time due to the bilinear pair operation. Both the proposed scheme and Scheme 2 have the lowest time complexity for proof validation.

5.2.4. Performance of DIST. According to our performance analysis, DIST can achieve data insertion, deletion, and modification with logarithmic time complexity. To validate our analysis, we implemented DIST in C++ and compared its performance with array and linked list. The performance of a dynamic operation array is consistent with that of an array after frequent data dynamics. We set the maximum number of DIST's two-layer pointer to 1000 and the maximum layer of DIST to 6 and conducted experiments involving 100 insertion, deletion, update, and search operations of different numbers of data blocks. The results of these experiments are presented in Table 5. From the table, we observed that the time required for inserting and deleting data using an array and the linked list has a small difference when the data volume is 10^6. The reason for this is that when performing insertion and deletion operations, an array requires subsequent data to be moved, whereas a linked list requires searching from the head node. However, DIST supports binary search and partial update, which allows for more efficient dynamic data operations. The time cost of DIST is approximately 1/500 of both. Furthermore, unlike an array and a linked list, where the time cost is proportional to the number of data blocks, the time cost of DIST increases logarithmically with the number of data blocks.

In summary, the proposed scheme in this paper outperforms the other three comparison schemes in terms of storage, communication overhead, and data dynamic efficiency. The overall time consumption of the proposed protocol is significantly lower than that of Schemes 1 and 3 and slightly better than Scheme 2. The key factors affecting the calculation time are the tag calculation and proof generation. Schemes 1 and 3 use the RSA and BLS signature algorithms, respectively, which involve exponentiation and bilinear pair operations, thus leading to higher computational complexity. The proposed scheme, however, is based on basic algebraic operations, and the most complex calculations only involve hash operations, making its performance similar to Scheme 2 (PRF-based PDP protocol). Our proposed DIST data structure provides nearly 500 times faster data dynamic operations compared to array and linked list, resulting in a significant improvement in data dynamic efficiency.

6. Conclusion

In this paper, we first present a novel cloud storage auditing protocol named AB-DPDP, which is based on an algebraic equation introduced in Theorem 1. We construct two secure cloud storage frameworks for this protocol, namely private verification and public verification. Then, to support dynamic data auditing, we present a dynamic index skip table data structure to support more efficient data dynamic operations. Next, we have proven the correctness and security of the protocol according to the security model. Finally, through a comprehensive evaluation, including theoretical analysis and experimental simulations, we verify the superiority of the AB-DPDP protocol and DIST. Compared to schemes based on signature algorithms, the AB-DPDP protocol offers not only the lowest storage and communication overheads but also reduces the time of tag computation by over 80% and the time of integrity verification by 70% for large files above 1 GB. Furthermore, when dealing with data volumes in the millions, the data dynamic efficiency of DIST improves by over 500 times compared to traditional index-transforming data structures. The AB-DPDP protocol is a highly suitable choice for cloud storage environments that require frequent data dynamic operations and strong data verification. In future research, we aim to explore the integration of identity-based cryptography to further reduce the overhead associated with key management.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2021YFB3101100), National Natural Science Foundation of China (No. U20A20177, 62272348, U22B2022), Key R&D Plan of Hubei Province (No. 2021BAA025), Industry-University-Research Innovation Fund for Chinese Universities (No. 2021FNA04004), and Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)(No. GML-KF-22-07).

References

[1] I. Gupta, A. K. Singh, C. Lee, and R. Buyya, "Secure data storage and sharing techniques for data protection in cloud environments: a systematic review, analysis, and future directions," *IEEE Access*, vol. 10, pp. 71247–71277, 2022.

[2] B. Grobauer, T. Walloschek, and E. Stöcker, "Understanding cloud computing vulnerabilities," *IEEE Security & privacy*, vol. 9, pp. 50–57, 2011.

[3] T. Wu, G. Yang, Y. Mu, R. Chen, and S. Xu, "Privacy-enhanced remote data integrity checking with updatable timestamp," *Information Sciences*, vol. 527, pp. 210–226, 2020.

[4] G. Ateniese, R. C. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds., ACM, Alexandria, Virginia, USA, pp. 598–609, 2007.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology*, vol. 26, pp. 442–483, 2013.

[6] J. Liu, K. Huang, H. Rong, H. Wang, and M. Xian, "Privacy-preserving public auditing for regenerating-code-based cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, pp. 1513–1528, 2015.

[7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *4th International ICST Conference on Security and Privacy in Communication Networks, SECURECOMM 2008*, A. Levi, P. Liu, and R. Molva, Eds., p. 9, ACM, Istanbul, Turkey, 2008.

[8] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 847–859, 2011.

[9] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 2402–2415, 2017.

[10] R. Rabaninejad, M. Ahmadian-Attari, M. R. Asaar, and M. R. Aref, "A lightweight auditing service for shared data with secure user revocation in cloud storage," *IEEE Transactions on Services Computing*, vol. 15, pp. 1–15, 2022.

[11] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Transactions on Cloud Computing*, vol. 9, pp. 923–937, 2021.

[12] Y. G. Ramaiah and G. V. Kumari, "Complete privacy preserving auditing for data integrity in cloud computing," in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013/11th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA-13/12th IEEE International Conference on Ubiquitous Computing and Communications*, pp. 1559–1566, IEEE Computer Society, Melbourne, Australia, July 2013.

[13] C. Hahn, H. Kwon, D. Kim, and J. Hur, "Enabling fast public auditing and data dynamics in cloud services," *IEEE Transactions on Services Computing*, vol. 15, pp. 2047–2059, 2022.

[14] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, pp. 297–319, 2004.

[15] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 78–88, 2017.

[16] J. Zhang, Y. Yang, Y. Chen, and F. Chen, "A secure cloud storage system based on discrete logarithm problem," in *Proceedings of the 25th IEEE/ACM International Symposium on Quality of Service, IWQoS 2017*, pp. 1–10, IEEE, Vilanova i la Geltrú, Spain, June 2017.

[17] Y. Yang, Y. Chen, and F. Chen, "A compressive integrity auditing protocol for secure cloud storage," *IEEE/ACM Transactions on Networking*, vol. 29, pp. 1197–1209, 2021.

[18] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference*, B. S. K. Jr., Ed., Springer, Santa Barbara, California, USA, pp. 112–131, 1997.

[19] A. Juels and S. K. Burton Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds., ACM, Alexandria, Virginia, USA, pp. 584–597, 2007.

[20] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds., ACM, Chicago, Illinois, USA, pp. 213–222, 2009.

[21] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security*, vol. 17, pp. 1–29, 2015.

[22] E. Esiner, A. Kachkeev, S. Braunfeld, A. Küpçü, and Ö. Özkasap, "Flexdpdp: flexlist-based optimized dynamic provable data possession," *ACM Transactions on Storage*, vol. 12, 2016.

[23] C. Liu, J. Chen, L. T. Yang et al., "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2234–2244, 2014.

[24] Y. Zhu, H. Wang, Z. Hu, G. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC)*, W. C. Chu, W. E. Wong, M. J. Palakal, and C. Hung, Eds., ACM, Tai-Chung, Taiwan, pp. 1550–1557, 2011.

[25] W. Guo, H. Zhang, S. Qin et al., "Outsourced dynamic provable data possession with batch update for secure cloud storage," *Future Generation Computer Systems*, vol. 95, pp. 309–322, 2019.

[26] H. Jin, H. Jiang, and K. Zhou, "Dynamic and public auditing with fair arbitration for cloud data," *IEEE Trans. Cloud Comput*, vol. 6, pp. 680–693, 2018.

[27] H. Tian, Y. Chen, C. Chang et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Serv. Comput*, vol. 10, pp. 701–714, 2017.

[28] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol. 64, pp. 3293–3303, 2015.

[29] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Information Sciences*, vol. 472, pp. 223–234, 2019.

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

[31] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020.

[32] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pp. 525–533, IEEE, San Diego, CA, USA, March 2010.

[33] H. Liu, P. Zhang, and J. Liu, "Public data integrity verification for secure cloud storage," *Journal of Networks*, vol. 8, pp. 373–380, 2013.

[34] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, pp. 114–121, 2014.

[35] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, pp. 608–619, 2020.

[36] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, pp. 72–83, 2019.

[37] A. Javadpour, A. K. Sangaiah, P. Pinto et al., "An energy-optimized embedded load balancing using DVFS computing in cloud data centers," *Computer Communications*, vol. 197, pp. 255–266, 2023.

[38] A. K. Sangaiah, A. Javadpour, F. Ja'fari, P. Pinto, W. Zhang, and S. Balasubramanian, "A hybrid heuristics artificial intelligence feature selection for intrusion detection classifiers in cloud of things," *Cluster Computing*, vol. 26, pp. 599–612, 2023.