Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5721917, 12 pages
https://doi.org/10.1155/2022/5721917

Research Article

Efficient Data Integrity Auditing Supporting Provable Data Update for Secure Cloud Storage

Changsong Yang[1,2] **Bowen Song**[1] **Yong Ding**[1,3] **Jiangtao Ou**[4] **and Chengyuan Fan**[4]

[1]*Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*Guangxi Cooperative Innovation Center of Cloud Computing and Big Data, Guilin University of Electronic Technology, Guilin 541004, China*
[3]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China*
[4]*AI Sensing Technology, Chancheng District, Foshan 528000, China*

Correspondence should be addressed to Changsong Yang; csyang@guet.edu.cn

Received 13 January 2022; Revised 14 February 2022; Accepted 15 March 2022; Published 26 March 2022

Academic Editor: Junjuan Xia

Cloud storage, an economically attractive service offered by cloud service providers (CSPs), has attracted a large number of tenants. However, because the ownership and management of outsourced data are separated, outsourced data faces a lot of security challenges, for instance, data security, data integrity, data update, and so on. In this article, we primarily investigate the problem of efficient data integrity auditing supporting provable data update in cloud computing environment. Subsequently, on the basis of the Merkel sum hash tree (MSHT), we introduce an efficient outsourced data integrity auditing scheme. Our designed scheme could synchronously meet the requirements of provable data update and data confidentiality without dependency on a third authority. At the same time, the numerical analysis shows that the computing complexity logarithmically grows with the number of outsourced subfiles. Finally, a prototype implementation is developed to simulate our designed scheme and measure its performance. The consequences of experiments present that compared with some previous solutions, our designed scheme has much more attractive practicality and higher efficiency in practical applications.

1. Introduction

With the fast development of computer and network technology, the total volume of digital data shows an exponential growth tendency [1]. The investigation report shows that there were 5200 GB digital data for everyone on average in 2020 [2]. However, the storage resources of tenants are so limited that cannot preserve such large-scale data. Therefore, massive data storage would become a challenging problem for resource-constrained tenants. Fortunately, cloud storage offers a potential solution to handle the issue of massive data storage and management. By embracing cloud storage services, tenants could upload their data to the cloud data center, thus efficaciously reducing the local memory space and computational cost [3]. Because of these attractive advantages, a growing number of tenants prefer to employ cloud storage services. According to the report of Cisco, there were 3.6 billion Internet consumers at the beginning of 2020. At the same time, about 55% of Internet consumers embraced cloud storage [4].

In cloud storage, the management of outsourced data is separated from its ownership [5]. Therefore, tenants would lose the physical management of their outsourced data, thus cannot directly perform any operations over the outsourced data [6]. In other words, the cloud data center performs all operations over the outsourced data. Nevertheless, the cloud data center is not fully reliable, and it might not honestly perform these operations according to the tenant's commands. As a result, although cloud storage has huge advantages, it inevitably faces plenty of serious problems [7], for instance, data confidentiality, data integrity, data update, etc. If these problems, especially data integrity, are not solved well, it will greatly prevent the public from accepting and employing cloud storage services [8].

To guarantee outsourced data integrity and availability, plenty of solutions have been presented [9–11]. However, there still exist some problems which need to be solved solidly. First, most of the previous solutions are based on proof of retrievability (PoR) technology or provable data possession (PDP) technology, whose computing complexity grows approximatively linearly with the scale of outsourced files. Hence, the efficiency is not attractive if the scale of outsourced file keeps growing [12]. Second, lots of the previous works require to depend on a third authority (TA), whose security and stability are particularly troubling. On the one hand, because of the expensive overhead, the third authority would refuse to provide services, which will cause service interruption. On the other hand, the third authority might be attacked by the adversary, and it cannot resist the commands of the government, which both would trigger privacy leakage. Last but not least, although some of the previous solutions concurrently accomplish data confidentiality, data integrity, and data update in some special scenarios, the practicability and universality are so limited that they cannot be applied to large-scale data outsourcing scenarios.

From the above analysis, we can easily discover that there are still some issues in the previous solutions. Further, there are few solutions that can simultaneously achieve data integrity auditing and dynamic data update efficiently without dependency on a third authority. As a consequence, the main motivation of this paper is to design a novel scheme to simultaneously solve the above problems. Specifically, we aim to make use of Merkle sum hash tree (MSHT) to design an efficient outsourced data integrity auditing scheme supporting provable data update without interacting with a third authority. As a consequence, the main contributions of this article are the two folds as below.

(i) We use MSHT to establish a high-performance outsourced data integrity verification scheme, which simultaneously supports provable outsourced data update. Specifically, the designed scheme can not only ensure the data integrity and usability but also meet the requirement of block-based data update, by which tenants can efficaciously update the outsourced data and check the update result without retrieving the data. At the same time, the designed scheme can realize the confidentiality of outsourced data, thus preventing the leakage of privacy information

(ii) Our designed scheme could achieve the anticipant functionality objectives without dependency on a third authority. In the meantime, under the random oracle model, we conduct a detailed security analysis to prove that the designed scheme is secure. Finally, a prototype implementation is developed to simulate the proposed scheme and perform the efficiency evaluation, which directly presents the practicability of the designed scheme

1.1. Related Works. Data integrity auditing has already been investigated in the past several decades both in academic and industrial, resulting in a rich body of literature [13–16]. In 2007, Ateniese et al. [17] first designed a PDP model, in which they utilized random sampling to realize data integrity auditing efficiently. In the meantime, they put forward two detailed PDP schemes which were provable secure and efficient. In order to achieve remote data possession checking (RDPC), Chen [18] utilized algebraic signatures to design a possession auditing scheme for remote data. In her designed scheme, the communication overhead was constant in challenge and response protocols, which could improve efficiency. However, Yu et al. [19] found that the previous protocol [18] was not able to prevent replay attacks and deletion attacks. At the same time, they proposed a novel remote cloud data possession auditing scheme, which could effectively resist deletion attack and replay attack. Li et al. [20] studied the challenge of multiple copies of data integrity auditing, in which multiple copies of data were maintained by multiple cloud data centers. After that, they presented a novel data integrity auditing protocol, which could improve efficiency by checking all copies at one time. Wang [21] designed a novel model called identity-based distributed PDP (ID-DPDP). At the same time, he utilized bilinear pairings to present a concrete ID-DPDP protocol for multicloud storage scenario. His proposed scheme could support private validation, delegated validation, and public validation. In order to improve the efficiency, Chang et al. [22] designed an identity-based PDP protocol, in which they adopted a novel hash function to save the communication overhead.

In 2019, Fan et al. [23] designed an identity-based aggregate signature (SIBAS) and took it as the secure integrity validation protocol over cloud data, which could synchronously realize reliable key management through Shamir's threshold protocol. In 2021, Shen et al. [24] presented a novel concept named data integrity reliable without private key storage and designed a concrete solution, in which they utilized the advantages of biometric data to remove the hardware token. Lu et al. [25] utilized hyperledger fabric to build a secure data integrity validation protocol with scalability. Zhang et al. [26] put forward a blockchain-based cloud data integrity verification protocol, which could support public verifiability. In their scheme, all the verification results would be recorded into a transaction that was time-sensitive. Li and Zhang [27] designed a certificate-based integrity auditing protocol, which was provable secure under the random oracle model. Moreover, their scheme required constant computational overhead to generate a verification tag for a data block. Nevertheless, the above solutions could not realize data dynamic operations, such as data update.

In order to simultaneously achieve data integrity auditing and data dynamic update, Liu et al. [28] presented an improved dynamic PDP (DPDP) protocol, which, respectively, utilized tags and hash values to guarantee data integrity and tag integrity. Chen and Curtmola [29] designed a remote DPDP scheme, which could achieve data integrity auditing and data update with constant storage overhead for client. Barsoum and Hasan [30] presented a PDP protocol for multi-backup dynamic cloud data, which was able to support full data dynamics. In 2015, Esiner et al. [31] utilized

flexlist to design an optimized DPDP scheme, which was able to hand plenty of updates simultaneously, thus improving efficiency. Yuan et al. [32] studied the integrity auditing for dynamic multi-replica data that was maintained by multiple cloud data centers, and they presented a solution to hand this problem. In their protocol, the modular exponentiation was replaced by the vector dot products, thus effectively saving computational resources. In 2021, Yu et al. [33] used indexed Merkle hash tree (IMHT) to design an efficient dynamic data auditing protocol, which could support fully dynamic operations. To achieve batch verification, Guo et al. [34] utilized implicitly indexed balanced Merkle tree (IBMT) to put forward a dynamic proof of data possession and replication (DPDPR) protocol. Their proposed solution greatly saved computational and communication resources. However, their solution required introducing a TPA to complete cloud data integrity verification.

Besides PDP, plenty of authentication data structures also can achieve integrity auditing and support data dynamic operations. He et al. [35] utilized permission-based signature and blockless Merkle tree to present a dynamic group-oriented PDP protocol, and they claimed that their protocol was an important phase in establishing efficient multi-writer cloud storage systems. Chen et al. [36] designed a novel PDP model based on blockchain, which might be utilized to achieve distributed cloud storage framework. Subsequently, they designed a concrete decentralized PDP scheme by utilizing multi-replica storage tricks, which supported dynamic operations over outsourced data. Recently, Guo et al. [37] connected multi-leaf-authenticated and rank-based Merkle tree to realize integrity checking and batch update over outsourced data in secure cloud storage. Chen et al. [38] utilized vector commitment (VC) to build a novel verifiable database (VDB) model supporting replacement update, which could resist forward automatic update attack. Chen et al. [39] provided an improved VDB scheme, which utilized committed invertible Bloom filter to realize full dynamic operations. Nevertheless, their schemes required to perform plenty of bilinear pairing computations, resulting in expensive computational overhead.

1.2. Organization. We organize the rest of this paper as below. Section 2 describes the preliminary of MSHT, which will be used to establish our novel scheme. In Section 3, we introduce the problem statements and then introduce the proposed scheme in Section 4. After that, the scheme analysis, including security analysis, functionality contrast, and computational complexity analysis is introduced in Section 5. Next, we implement our designed scheme and evaluate its efficiency in Section 6. At last, Section 7 simply concludes this paper.

2. Preliminaries

MSHT was originally designed by Miao et al. in 2018 [40], which could be regarded as a development of the traditional MHT [41]. Similar to MHT, MSHT can also verify the data integrity of any subset [42]. However, there are two differences between MHT and MSHT. On the one hand, all leaves of MSHT could manage many data blocks, but each leaf of MHT could merely save one data block. On the other hand, the amount of data blocks managed by the leaf is used as input to generate the hash value, as demonstrated in Figure 1. Subsequently, the Merkle root node H_R contains all of the data blocks in the given set, and a signature Sig_R can be generated on the hash value H_R through a provable secure signature protocol.

Similar to MHT, MSHT also relies on the auxiliary validation information φ to achieve data integrity auditing. Auxiliary validation information φ contains the hash values and the data block numbers of the sibling nodes on the path from the verified leaf node to the Merkle root node [43]. For example, to verify the integrity of data block $f_{3,1}$, the auxiliary validation information is $\varphi_3 = ((h_{14}, 1), (h_{21}, 2))$. Then, the verifier can recompute a novel Merkle root node H_R' by utilizing the auxiliary validation information φ_3 and compare H_R' with H_R. At the same time, the verifier, audits the correctness of signature Sig_R. If and only if H_R is equal to H_R' and Sig_R is a correct signature on Merkle root node H_R', will the verifier believe that $f_{3,1}$ is intact.

3. Problem Statements

We present the problem statements (i.e., system framework, technology challenges, and design goals) in this section.

3.1. System Framework. We study the problem of high efficiency data integrity checking supporting provable data update for secure cloud storage, whose system framework consists of two participants: a tenant and a cloud data center, as demonstrated in Figure 2.

3.1.1. Tenant. The tenant is equipped with restricted storage capacities, so he/she could not save and manage massive data by himself/herself in local storage mediums. Therefore, the tenant prefers to upload his/her massive data to the remote cloud data center. Later, the tenant wants to utilize a few novel data blocks to replace some old data blocks for efficiently updating the outsourced data set. Since the tenant lacks trust in the cloud data center, the tenant would want to audit the integrity of the outsourced data and the result of the data update operation.

3.1.2. Cloud Data Center. The cloud data center connects plenty of dispersive physical disks, network resources, and computational devices by the Internet and establishes a sharing resource pool, thus providing the tenant with convenient and efficient storage services. Meanwhile, since the ownership outsourced data is separated from its management, the cloud data center executes outsourced data update operations for the tenant and returns related proofs to convince the tenant that the outsourced data blocks have been correctly updated according to the tenant's commands.

3.2. Security Challenges. The following three security challenges must be seriously considered and solidly solved in our proposal.

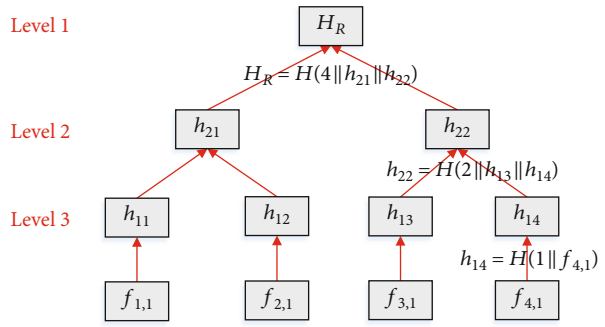

Figure 1: Basic structure of MSHT.

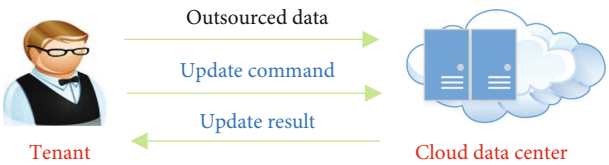

Figure 2: System framework of the designed scheme.

3.2.1. Privacy Leakage. Outsourced data usually contains some privacy information of tenant. In cloud storage, the cloud manager is so curious that peeps at the outsourced data for finding some privacy information of the tenant. Meanwhile, external attackers (such as hackers) might illegally access the outsourced data to dig the privacy information. As a consequence, privacy leakage is a severe challenge that should be seriously considered.

3.2.2. Data Pollution. Because of the following three reasons, outsourced data might be polluted. First, data loss would be caused by software failure, hardware breakdown, and erroneous operations of the cloud manager. Second, the cloud data center may maliciously remove a few outsourced data blocks which are seldom utilized in order to save storage overhead. Third, hacker might maliciously target the cloud data center, causing damage to the outsourced data.

3.2.3. Dishonest Data Update. The cloud data center must run some complex protocols or calculations in order to achieve outsourced data update. In consequence, the cloud data center has to cost some expensive computational overhead and some additional storage capacities, which is undesired for the cloud data center. Therefore, the cloud data center may not honestly execute the tenant's command to update the outsourced data.

3.3. Security Goals. In the designed scheme, we require to meet three requirements as follows, including data confidentiality, data integrity auditing, and provable data update.

3.3.1. Data Confidentiality. Data confidentiality refers to keeping the tenant's privacy information included in the outsourced data secret. In consequence, to avoid the leakage of privacy information, the tenant should encrypt the outsourced file by utilizing a secure encryption protocol before uploading.

3.3.2. Data Integrity Auditing. Data integrity is defined that the cloud data center saves and manages the outsourced data integrally. If malicious data pollution has occurred in outsourced data blocks, the tenant can effectively discover the data pollution through periodically performing a data integrity auditing operation.

3.3.3. Provable Data Update. Provable data update refers to that the cloud data center must sincerely perform the tenant's outsourced data update command and honestly return related evidence. The tenant can check the returned evidence to detect malicious acts if the cloud data center did not honestly update the outsourced data.

4. Our Scheme

We establish a high-efficiency data integrity auditing scheme, which can also realize data confidentiality and verifiable data update in secure cloud storage environment. Specifically, the proposed scheme contains five main phases: initialization, data preprocessing, data outsourcing, data integrity auditing, and data update.

4.1. Initialization. In this phase, the tenant achieves registration on the cloud data center. Meanwhile, some related public parameters and public/privacy key pairs are generated.

4.1.1. Tenant Registration. Before employing the cloud data center's data storage services, the tenant must become a legitimate customer of the cloud service providers. Hence, the tenant has to register and complete the identity authentication on the cloud data center. Then, the tenant becomes a legitimate customer of the cloud data center and obtains a unique identity number ID. As a consequence, the tenant can employ the cloud data center's data storage services directly.

4.1.2. Parameter/Key Generation. Firstly, the public/private key pair (PK_C, SK_C) for the elliptic curve digital signature algorithm (ECDSA) is generated by the cloud data center. Then, the cloud data center publishes the public key PK_C and keeps the privacy SK_C so confidential that nobody else can obtain it. Similarly, the tenant computes public/privacy key pair (PK_T, SK_T). Subsequently, the tenant publishes the public key PK_T and keeps the privacy SK_T confidential. At the same time, the tenant picks an identifier n_f for the file F.

4.2. Data Preprocessing. This phase mainly achieves data encryption and ciphertext segmentation. The detailed process of data preprocessing is below.

4.2.1. Data Encryption. Generally, the outsourced file involves some privacy information which has to be kept confidential. Hence, before uploading the outsourced data to the cloud data center, the tenant should encrypt it to protect the privacy data. To be more specific, the tenant first computes a key $K = H(ID \| n_f \| SK_T)$, in which $H(\cdot)$ is a secure hash algorithm. After that, the tenant executes data encryption operation $f = Enc_K(F)$, in which Enc is a symmetric encryption scheme, and it is indistinguishable under chosen-plaintext attack (IND-CPA), and f is the corresponding ciphertext.

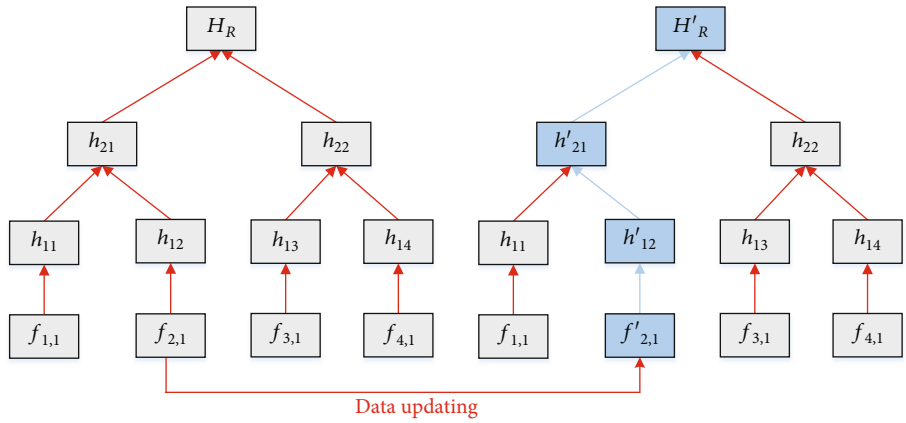

Figure 3: An illustration of data updating.

4.2.2. Ciphertext Segmentation. The tenant firstly divides the related ciphertext f into n subfiles $f = (f_1, f_2, \cdots, f_n)$. Subsequently, the tenant further splits each subfile into S data blocks $f_{i1}, f_{i2}, \cdots, f_{is}$, where $i = 1, 2, \cdots, n$. As a consequence, the tenant can obtain the outsourced data set $f = (f_1, f_2, \cdots, f_n) = \{f_{i,j}\}_{1 \le i \le n, 1 \le j \le S}$.

4.3. Data Outsourcing. In this phase, the outsourced data set is maintained through building an MSHT, and then, the whole tree is outsourced to the cloud data center. The details of data outsourcing are described as below.

4.3.1. Tree Building. The tenant uses the received data blocks to construct a Merkle sum hash tree *MSHT* that contains n leaf nodes $N_1, N_2, \cdots, N_n$. In *MSHT*, every leaf saves a subfile, i.e., leaf N_i saves subfile f_i. Subsequently, the tenant could acquire a Merkle root H_R. Meanwhile, the tenant computes a signature Sig_R for hash value H_R, where $Sig_R = Sign_{SK_T}(H_R)$, and *Sign* is the signature computation algorithm of ECDSA. Next, the tenant uploads the whole Merkle sum hash tree *MSHT* to the cloud data center, along with the Merkle root's signature Sig_R.

4.3.2. File Storage. On receipt of the outsourced data set f, the tree *MSHT*, and the signature Sig_R, the cloud data center audits the correctness of *MSHT*. To be more precise, the cloud data center rebuilds the tree by utilizing outsourced data set f and then acquires a novel hash value of the Merkle root H'_R. Subsequently, the cloud data center checks if equality $H_R = H'_R$ holds, meanwhile, confirms the correctness of the signature Sig_R. If and only if equality $H_R = H'_R$ holds and signature Sig_R is correct, will the cloud data center think the received tree *MSHT* is correct and then manage the outsourced file for the tenant by saving the tree *MSHT*. At last, the cloud data center returns success to the tenant to indicate the data storage result.

4.3.3. Data Deletion. After the outsourced file is successfully stored by the cloud data center, the tenant deletes all copies of outsourced file F and outsourced data set f for saving local storage capacities. Subsequently, the tenant merely stores the Merkle root H_R in local physical disks.

Table 1: Functionality contrast.

Scheme	Scheme [16]	Scheme [33]	Our scheme
Confidentiality	✓	✓	✓
Data integrity	✓	✓	✓
Data update	☒	✓	✓
Without TA	✓	☒	✓

4.4. Integrity Auditing. After the tenant uploads the outsourced data to the cloud data center, the data integrity auditing is periodically executed to ensure the integrity of outsourced data. Then, the details of data integrity auditing are below.

4.4.1. Information Retrieval. The tenant firstly discretionarily picks an integer i, where $1 \le i \le n$. Subsequently, the tenant gets the subfile $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{is})$ and its related auxiliary validation information φ_i back from the cloud data center.

4.4.2. Merkle Root Rebuilding. The tenant utilizes the auxiliary validation information φ_i and the subfile $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{is})$ to rebuild the Merkle root. Subsequently, the tenant can obtain a novel hash value H^*_R of Merkle root for contrasting with H_R. If equality $H_R = H^*_R$ does not hold, the tenant believes that the subfile $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{is})$ has been polluted; Otherwise, if equality $H_R = H^*_R$ holds, the tenant can believe the subfile $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{is})$ is intact.

4.5. Data Update. The tenant wants to utilize a few novel data blocks to replace some old outsourced data blocks, thus efficiently updating the outsourced file. Subsequently, the detailed steps of data update are below.

4.5.1. Command Generation. For simplicity, assume that the tenant hopes to update the outsourced data block $f_{k,p}$, which is stored by leaf node N_k, where $1 \le k \le n$ and $1 \le p \le S$. Subsequently, the tenant retrieves the subfile $f_k = (f_{k,1}, f_{k,2}, \cdots,$

Table 2: Computational complexity contrast.

Scheme	Scheme [16]	Scheme [33]	Our scheme
Data preprocessing	$nE + 2nH$	–	$1E + 1H$
Data outsourcing	$n(Exp + M)$	$n(Exp + H)$	$4nH + 1S + 1V$
Integrity auditing	$2P + nExp$	$2P + sExp + sH$	$k \log_2 nH$
Data updating	–	$\log_2 nH + sExp + 1S$	$2(S + V + q \log_2 nH)$

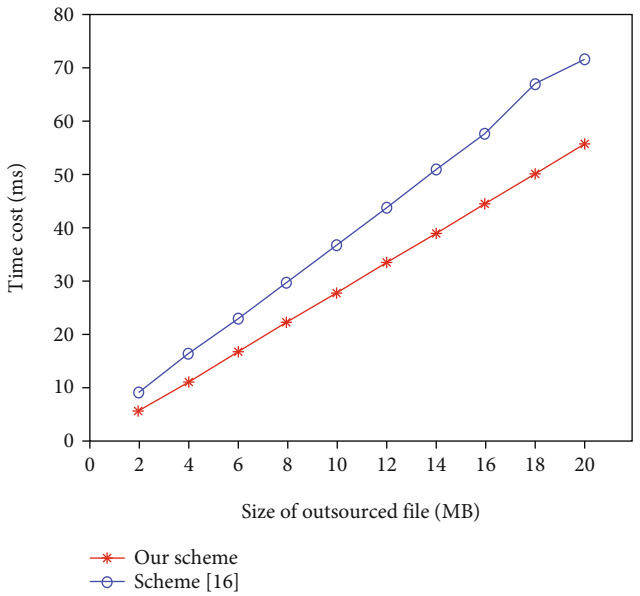

Figure 4: Time cost of data preprocessing.

$f_{k,S}$) from the cloud data center, as well as its auxiliary validation information φ_k. Meanwhile, the tenant generates a signature $Sig_T = \text{Sign}_{SK_T}(\text{update} \| k \| p \| T)$, where T represents the timestamp. Next, the tenant can further generate a data update command $UC = (\text{update}, k, p, T, Sig_T)$. Finally, the tenant transmits the data update order UC to the cloud data center, along with the novel data block $f'_{k,p}$.

4.5.2. Data Updating. When the cloud data center receives a data update command UC and a new data block $f'_{k,p}$, the cloud data center firstly audits the correctness of the data update command UC by signature checking. To be more specific, the cloud data center validates the signature Sig_T. If Sig_T is an invalid signature, the cloud data center returns failure; otherwise, if signature Sig_T is correct, the cloud data center replaces the old data block $f_{k,p}$ with the new data block $f'_{k,p}$. Meanwhile, the cloud data center obtains a new Merkle sum hash tree $MSHT'$, as presented in Figure 3. Subsequently, the cloud data center transmits the new Merkle root H_r, along with its corresponding signature Sig_r to the tenant, where $Sig_r = \text{Sign}_{SK_C}(H_r)$.

4.5.3. Result Checking. Upon receiving H_r and Sig_r from the cloud data center, the tenant could audit the data update result. Specifically, the tenant firstly utilizes the auxiliary validation information φ_k and the new subfile $f'_k = (f_{k,1}, \cdots, f'_{k,p}, \cdots, f_{k,S})$ to acquire a novel hash value H'_r of the Merkle root for contrast with H_r, where H_r is returned by the cloud data center. Meanwhile, the tenant validates the validity of the signature Sig_r. If and only if the equality $H_r = H'_r$ holds and signature Sig_r is correct, will the tenant think that the cloud data center has honestly executed the data update command.

5. Scheme Analysis

5.1. Security Proof. We would formally demonstrate that the novel designed scheme could achieve the anticipant design goals, including data confidentiality, data integrity, and provable data update.

5.1.1. Theorem 1: The Designed Scheme Could Achieve Data Confidentiality. Data confidentiality refers to that the privacy information included in outsourced data should not be disclosed. Generally, outsourced data should be encrypted with a safe encryption protocol to avoid the leakage of sensitive information. As a consequence, the encryption protocol directly determines the security of outsourced data. In our designed scheme, before storing the outsourced file on the cloud data center, the tenant encrypts the outsourced file with an IND-CPA secure symmetric encryption protocol, such as *AES*. Thus, the corresponding ciphertext IND-CPA is secure too. As a result, if the attacker is unable to obtain the corresponding decryption key, he/she will be unable to extract any plaintext information from the ciphertext with a nonzero probability in polynomial time [44, 45]. That is, in our designed scheme, no attacker can get any plaintext data from the ciphertext if the tenant keeps the encryption/decryption keys secret.

5.1.2. Theorem 2: The Designed Scheme Could Ensure Data Integrity. Data integrity is defined that the cloud data center has to save and manage the outsourced data honestly and prevent it from being polluted. Otherwise, the tenant could discover the malicious outsourced data destruction with an overwhelming probability. In our designed scheme, the tenant randomly chooses an integer i from Z_n. After that, the tenant retrieves the subfile $f_i = (f_{i,1}, f_{i,2}, \cdots, f_{i,S})$ from the cloud data center, as well as its auxiliary validation information φ_i. As a result, the tenant can recompute a novel hash value H^*_R of Merkle root for contrast with H_R. Note that in data outsourcing scenario, the scale of outsourced subfiles

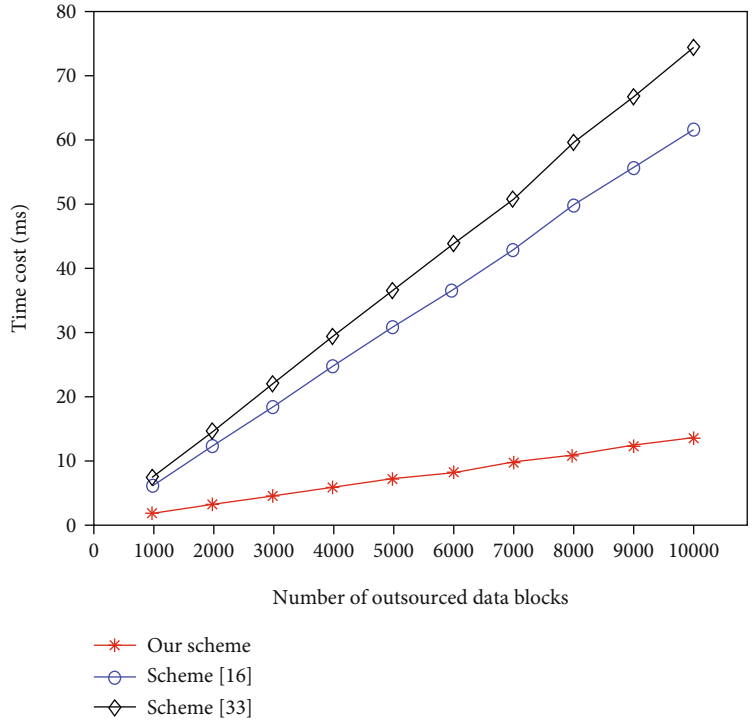

Figure 5: Time cost of data outsourcing.

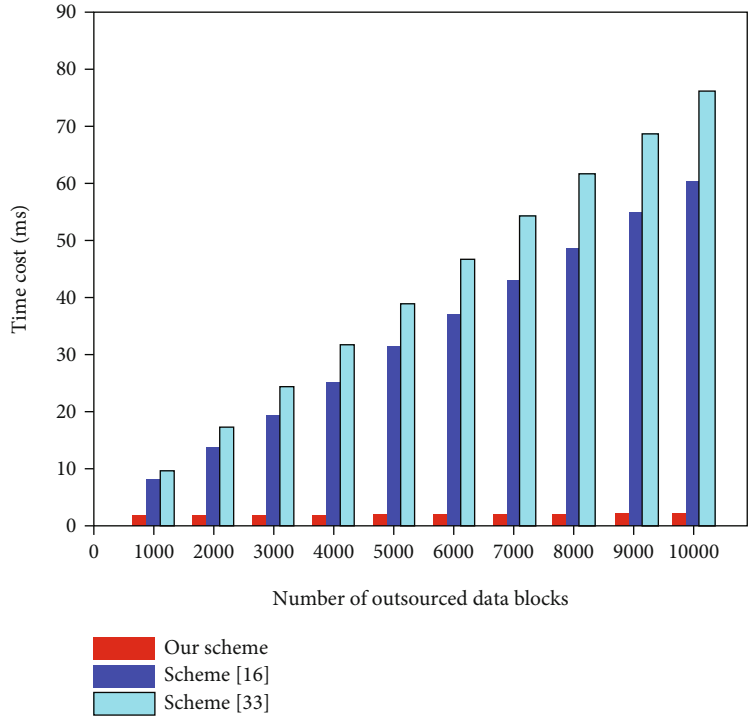

Figure 6: Time cost of integrity auditing.

is so large that the cloud data center could not correctly guess the chosen number i with nonnegligible probability in advance. Meanwhile, the hash function is collision-resistant and one-way. Therefore, if the outsourced data is polluted, in polynomial time, the cloud data center could not successfully forge a new data block to make equality $H_R^* = H_R$ hold with a nonnegligible probability. In other words, if and only if the outsourced data is intact, will

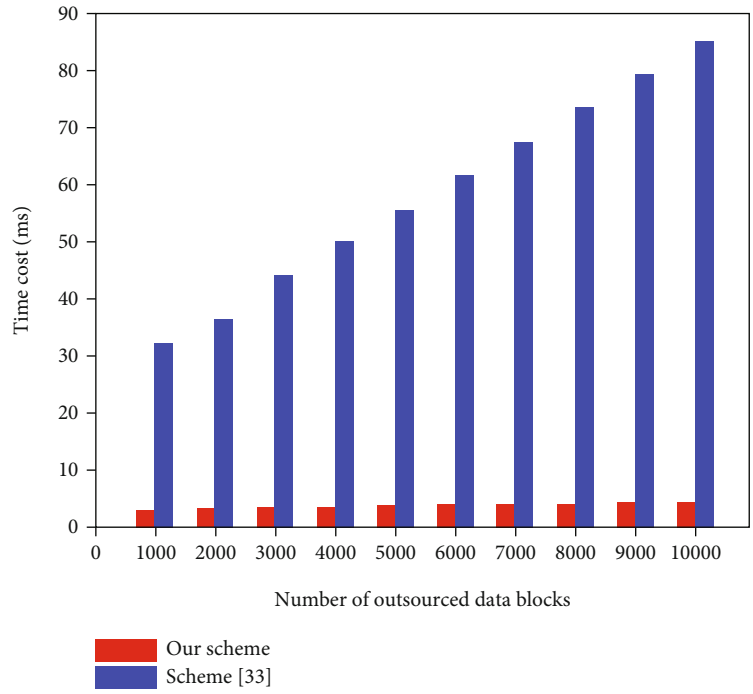

Figure 7: Time cost of data updating.

equality $H_R^* = H_R$ hold. That is, if and only if equality $H_R^* = H_R$ holds, will the tenant believe that the outsourced data is intact.

5.1.3. Theorem 3: The Designed Scheme Could Achieve Provable Data Update. Provable data update means the cloud data center sincerely updates the outsourced data under the tenant's permission. To guarantee the data update operation is executed under the tenant's permission, the cloud data center verifies whether the data update command DC is correct and valid by signature verification. Note that the data update command DC contains a digital signature Sig_T, which is generated by the tenant with the privacy key SK_T. As we know, the privacy key SK_T is kept so secret by the tenant that nobody else can obtain it. Further, only the tenant can compute the signature SK_T and generate the data update command DC. Therefore, the signature SK_T can be seen as evidence which proves that the tenant prefers to update the outsourced data set. In other words, if signature SK_T is valid, it can mean that the data update command is indeed generated by the tenant. Then, the data update operation is executed by the cloud data center under the tenant's permission.

Moreover, since the tenant lacks trust in the cloud data center, he/she would want to authenticate the result of data update to guarantee the data update command is honestly carried out by the cloud data center. Specifically, the tenant firstly checks the correctness of signature Sig_r. When signature Sig_r is correct, the tenant thinks that the Merkle root H_r is returned by the cloud data center without being tampering. Subsequently, the tenant utilizes the new subfile $f_k' = (f_{k,1}, \cdots, f_{k,p}', \cdots, f_{k,S})$ and the auxiliary validation information φ_k to recompute a novel Merkle root H_r'. Then, the tenant contrasts the new Merkle root H_r' with the Merkle root H_r that is returned by the cloud data center. Note that the chosen hash function is collision-resistant and one-way. Therefore, if the cloud data center did not update the outsourced data set honestly, in polynomial time, it is unable to efficiently forge the Merkle root H_r to make equality $H_r' = H_r$ hold with a nonnegligible probability. As a consequence, if equality $H_r' = H_r$ holds and signature Sig_r is correct, the tenant can believe that the data update command has been honestly performed by the cloud data center.

5.2. Functionality Contrast. We would compare the functionality of our designed scheme and two previous schemes [16, 33], as demonstrated in Table 1.

From Table 1, we can obviously get three findings as below. First, the three solutions all can ensure data integrity, which enables the cloud data center to maintain the outsourced file honestly. Meanwhile, they all utilize secure encryption algorithms to prevent privacy information from leakage, thus ensuring data confidentiality. Second, our designed scheme and the previous scheme [33] can achieve block-based data update, but the previous scheme [16] cannot support data update. Finally, the previous scheme [33] requires relying on a TA to realize data integrity checking and provable data update, while the other two solutions do not need to interact with a TA. Therefore, we can trust that the overall functionalities of our designed scheme are more appealing than the other two previous solutions [16, 33].

5.3. Numeric Analysis. We present the numeric analysis through computational complexity contrast. We first

introduce some symbols which would be utilized for the contrast. To be more specific, we adopt symbols E, H, Exp, and M to, respectively, represent encryption computation, hash calculation, modular exponentiation computation, and multiplication operation. Meanwhile, we utilize symbol S and symbol V to, respectively, represent signature computation and signature auditing. Moreover, symbol n represents the number of outsourced subfiles, symbol P represents bilinear pairings computation, symbol k denotes the number of audited data blocks, and symbol q denotes the number of updated data blocks. In scheme [33], we assume each data block would be further split into s sectors.

From Table 2, we obviously discover our designed scheme which merely requires to compute/audit some signatures and compute some hash values. However, the other two solutions [16, 33] need to perform some modular exponentiation calculations or bilinear pairing computations. Therefore, we can consider that our designed scheme has higher efficiency than the other two previous solutions [16, 33].

6. Experimental Results

We carry out a prototype system to simulate our designed solution. Meanwhile, we evaluate the computational performance of executing the primary computations in each process. All the experiments are carried out on a desktop with Unix system, Intel(R) Core(TM) i7-7700 CPU running at 8 GB random access memory and 3.6 GHz.

In the simulation experiments, the pairing-based cryptography library and the open secure socket layer library are used to simulate the related cryptographic algorithms. To be more specific, $SHA-1$ is chosen as the secure hash function and AES is chosen as the secure encryption algorithm. Moreover, we choose ECDSA as the signature algorithm. For simplicity, we ignore the communication cost and assume that in our designed solution, each subfile is divided into a data block. That is, we treat a subfile as a data block in the experiments.

6.1. Time Overhead of Data Preprocessing. This data preprocessing phase aims to complement data encryption and ciphertext segmentation. As a consequence, the running time overhead of data preprocessing is concerned with the scale of outsourced file and the quantity of outsourced subfiles. In the experiment, the number of outsourced subfiles is set to $n = 3000$ and the scale of outsourced file is increased from 2 MB to 20 MB. Subsequently, we measure the time overhead of data preprocessing, as presented in Figure 4.

Figure 4 clearly demonstrates that the time overhead of data preprocessing linearly increases with the scale of outsourced data approximatively. At the same time, our designed solution's growth rate is a little lower compared with that of the previous solution [16]. Moreover, we are able to discover that our designed solution requires less time than the previous solution [16] under the same size of outsourced file. For example, if the scale of outsourced data reaches 20 MB, our designed solution requires nearly 55.6 milliseconds, while the previous solution [16] costs about 71.5 milliseconds. As a result, although the data preprocessing phase is performed by the tenant, the data preprocessing is a single-use for a given outsourced file; meanwhile, it can be executed off-line. In consequence, we could trust that our designed solution is quite high-efficiency in the step of data preprocessing.

6.2. Time Overhead of Data Outsourcing. This data outsourcing phase aims to build a data structure for managing the outsourced file and upload the outsourced file to the cloud data center. Therefore, the running time overhead of data outsourcing is connected to the quantity of data blocks. For convenience, the quantity s is set to 500. At the same time, the quantity of data blocks is increased from 1000 to 10000. Subsequently, we provide the running time overhead, as presented in Figure 5.

Figure 5 shows that in data outsourcing phase, all the running time overhead of the three solutions roughly grows with the quantity of data blocks. In the meantime, our designed solution's growth rate is the lowest among the three solutions. Moreover, the other two solutions [16, 33] both require significantly more time than our designed solution. For instance, when the quantity of data blocks is 5000, the time overhead of our designed solution is nearly 7.0 milliseconds, but the time of the previous solution [16] is 30.8 milliseconds, and the time overhead of the previous solution [33] is approximately 36.3 milliseconds. Foreseeably, our designed solution would require the least time overhead to outsource the same quantity of data blocks. As a consequence, we could directly consider our designed solution is more efficient than the other two solutions [16, 33] in data outsourcing phase.

6.3. Time Overhead of Integrity Auditing. The data integrity checking process aims to check the integrity and availability to ensure the cloud data center sincerely saves and manages the outsourced data. Hence, the running time overhead of integrity auditing is connected to the quantity of outsourced data blocks. As a consequence, we let $k = 100$ and $s = n$ for convenience. Meanwhile, the quantity of outsourced data blocks is increased from 1000 to 10000. Subsequently, the running time cost of data integrity auditing is presented in Figure 6.

From Figure 6, we would directly discover that all the time cost of the three solutions grows with the quantity of outsourced data blocks, while our designed solution's growth rate is the lowest. This is because the time cost of our designed solution logarithmically increases with the quantity of outsourced data blocks, while the running time cost of the other two previous solutions [16, 33] linearly grows with the quantity of outsourced data blocks approximatively. Furthermore, our designed solution requires the least time cost, and the previous solution [33] requires the most time cost. For instance, if the quantity of outsourced data blocks is 5000, the running time cost of our designed solution is nearly 1.7 milliseconds, while the time overhead of the previous solution [16] is approximately 31.1 milliseconds, and the time of the previous solution [33] is about 38.9 milliseconds. Therefore, we could trust that the efficiency of our designed solution is higher than these of the other two previous solutions [16, 33].

6.4. Time Overhead of Data Updating. This data updating phase aims to utilize a few novel data blocks to replace the old data blocks, thus updating the outsourced data. Hence, the running time overhead of data updating is concerned with the quantity of outsourced subfiles. We let $q = 100$ and $s = n$ for convenience. Meanwhile, we grow the quantity of outsourced subfiles from 1000 to 10000. Subsequently, we measure the running time overhead, as presented in Figure 7.

From Figure 7, we can clearly discover that in data updating phase, the running time cost of both our designed solution and the previous solution [33] grows with the quantity of outsourced data blocks. At the same time, the time overhead of our designed solution shows a much lower growth rate compared with that of the previous solution [33]. Specifically, the running time cost of our designed solution shows a logarithmic growth tendency, while the time overhead of the previous solution [33] presents a linear growth tendency. Furthermore, our proposed solution requires a substantially less time overhead than the previous solution [33]. For instance, when the quantity of outsourced subfiles is 5000, the time overhead of our proposed solution is nearly 3.5 milliseconds. Nevertheless, the running time overhead of the previous solution [33] is 55.3 milliseconds. Therefore, we can trust that in data updating phase, our designed solution is equipped with much higher efficiency than the previous solution [33].

7. Conclusions

In this paper, we study the issue of outsourced data integrity checking with provable data update in cloud computing environment. In the meantime, we utilized MSHT to present an efficient integrity auditing scheme over outsourced data, which could also achieve block-based dynamic data update. Specifically, the tenant could permanently delete the local backups from the physical medium after the file was stored on the cloud data center. Meanwhile, the tenant could periodically perform data integrity and availability auditing to ensure the cloud data center sincerely stored and managed the outsourced data. Subsequently, the tenant was able to utilize a few novel data blocks to replace the old outsourced data blocks to achieve data update. Because the tenant lacks trust in the cloud data center, he/she could audit the result of data update. The security analysis formally proved that our proposed scheme could realize anticipant design goal without dependency on a third authority. Finally, the performance comparison and experiment results showed that our designed scheme was much more high-efficient.

In our designed scheme, we solved the problem of efficient data integrity auditing with provable data replacement update. However, in some specific scenarios, tenants not only want to audit the data integrity and update the data but also want to add a few novel data blocks and remove some old data blocks. Therefore, in the future, we aim to study the design of efficient data auditing scheme supporting all dynamic operations, thus simultaneously achieving data integrity checking, dynamic data insertion, verifiable data update, and secure data deletion.

Data Availability

The data utilized to sustain the contributions of this research are all contained in the paper. Thence, there is no need of more data collection.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to sincerely thank the anonymous referees for their very valuable time. Meanwhile, this work is supported by the Science and Technology Program of Guangxi (AD20297028), the Natural Science Foundation of Guangxi (2020GXNSFBA297132), the National Key R&D Program of China (2020YFB1006003), the Guangdong Key R&D Program (2020B0101090002), the National Natural Science Foundation of China (61772150), and the Innovation Project of GUET Graduate Education (no. 2021YCXS063).

References

[1] C. Yang, F. Zhao, X. Tao, and Y. Wang, "Publicly verifiable outsourced data migration scheme supporting efficient integrity checking," *Journal of Network and Computer Applications*, vol. 192, 2021.

[2] T. Wang, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 3–12, 2018.

[3] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2062–2074, 2018.

[4] C. Yang, X. Tao, F. Zhao, and Y. Wang, "Secure data transfer and deletion from counting bloom filter in cloud computing," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 273–280, 2020.

[5] J. Li, J. Wu, and L. Chen, "Block-secure: blockchain based scheme for secure P2P cloud storage," *Information Sciences*, vol. 465, pp. 219–231, 2018.

[6] C. Yang, X. Chen, and X. Yang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of Network and Computer Applications*, vol. 103, pp. 185–193, 2018.

[7] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

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

[9] P. Wei, D. Wang, Y. Zhao, S. K. S. Tyagi, and N. Kumar, "Blockchain data-based cloud data integrity protection mechanism," *Future Generation Computer Systems*, vol. 102, pp. 902–911, 2020.

[10] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2021.

[11] Y. Yu, Y. Li, B. Yang, W. Susilo, G. Yang, and J. Bai, "Attribute-based cloud data integrity auditing for secure outsourced storage," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 377–390, 2020.

[12] C. Yang, X. Tao, S. Wang, and F. Zhao, "Data integrity checking supporting reliable data migration in cloud storage," in *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 615–626, Springer, 2020.

[13] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 72–83, 2019.

[14] G. Cui, Q. He, B. Li et al., "Efficient verification of edge data integrity in edge computing environment," *IEEE Transactions on Services Computing*, 2022.

[15] J. Lin, W. Yu, N. Zhang, X. Yang, and L. Ge, "Data integrity attacks against dynamic route guidance in transportation-based cyber-physical systems: modeling, analysis, and defense," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8738–8753, 2018.

[16] C. Yang, J. Wang, X. Tao, and X. Chen, "Publicly verifiable data transfer and deletion scheme for cloud storage," *International Journal of Distributed Sensor Networks*, vol. 15, 458 pages, 2019.

[17] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *in the 14th ACM conference on Computer and communications security*, pp. 598–609, Alexandria, Virginia, USA, 2007.

[18] L. Chen, "Using algebraic signatures to check data possession in cloud storage," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1709–1715, 2013.

[19] Y. Yu, Y. Zhang, J. Ni, M. H. Au, L. Chen, and H. Liu, "Remote data possession checking with enhanced security for cloud storage," *Future Generation Computer Systems*, vol. 52, pp. 77–85, 2015.

[20] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Transactions on Cloud Computing*, vol. 10, 2022.

[21] H. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 328–340, 2015.

[22] J. Chang, B. Shao, Y. Ji, and G. Bian, "Efficient identity-based provable multi-copy data possession in multi-cloud storage, revisited," *IEEE Communications Letters*, vol. 24, no. 12, pp. 2723–2727, 2020.

[23] Y. Fan, X. Lin, G. Tan, Y. Zhang, W. Dong, and J. Lui, "One secure data integrity verification scheme for cloud storage," *Future Generation Computer Systems*, vol. 96, pp. 376–385, 2019.

[24] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1408–1421, 2021.

[25] N. Lu, Y. Zhang, W. Shi, S. Kumari, and K.-K. R. Choo, "A secure and scalable data integrity auditing scheme based on hyperledger fabric," *Computers & Security*, vol. 92, 2020.

[26] 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, no. 3, pp. 923–937, 2021.

[27] Y. Li and F. Zhang, "An efficient certificate-based data integrity auditing protocol for cloud-assisted WBANs," *IEEE Internet of Things Journal*, 2021.

[28] F. Liu, D. Gu, and H. Lu, "An improved dynamic provable data possession model," in *in 2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pp. 290–295, Beijing, China, 2011.

[29] B. Chen and R. Curtmola, "Robust dynamic provable data possession," in *in 2012 32nd International Conference on Distributed Computing Systems Workshops*, pp. 515–525, Macau, China, 2012.

[30] F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 485–497, 2015.

[31] E. Esiner, A. Kachkeev, S. Braunfeld, A. Küpçü, and Ö. Özkasap, "FlexDPDP," *ACM Transactions on Storage*, vol. 12, no. 4, pp. 1–44, 2016.

[32] Y. Yuan, J. Zhang, and W. Xu, "Dynamic multiple-replica provable data possession in cloud storage system," *IEEE Access*, vol. 8, pp. 120778–120784, 2020.

[33] H. Yu, Z. Yang, M. Waqas et al., "Efficient dynamic multi-replica auditing for the cloud with geographic location," *Future Generation Computer Systems*, vol. 125, pp. 285–298, 2021.

[34] W. Guo, S. Qin, F. Gao et al., "Dynamic proof of data possession and replication with tree sharing and batch verification in the cloud," *IEEE Transactions on Services Computing*, 2020.

[35] K. He, J. Chen, Q. Yuan, S. Ji, D. He, and R. Du, "Dynamic group-oriented provable data possession in the cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, p. 1, 2021.

[36] R. Chen, Y. Li, Y. Yu, H. Li, X. Chen, and W. Susilo, "Blockchain-based dynamic provable data possession for smart cities," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4143–4154, 2020.

[37] 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.

[38] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 546–556, 2015.

[39] X. Chen, H. Li, J. Li et al., "Publicly verifiable databases with all efficient updating operations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 12, pp. 3729–3740, 2021.

[40] M. Miao, J. Ma, X. Huang, and Q. Wang, "Efficient verifiable databases with insertion/deletion operations from delegating polynomial functions," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 2, pp. 511–520, 2018.

[41] L. Rao, H. Zhang, and T. Tu, "Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 451–463, 2020.

[42] C. Yang, Y. Liu, and X. Tao, "Assure deletion supporting dynamic insertion for outsourced data in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 9, 2020.

[43] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, and J. Liu, "A position-aware Merkle tree for dynamic cloud data integrity verification," *Soft Computing*, vol. 21, no. 8, pp. 2151–2164, 2017.

[44] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang, "Assured data deletion with fine-grained access control for fog-based industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4538–4547, 2018.

[45] C. Yang, Y. Liu, X. Tao, and F. Zhao, "Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing," *IEEE Access*, vol. 8, pp. 99393–99403, 2020.