Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5758778, 10 pages
https://doi.org/10.1155/2022/5758778

Research Article

Provable Cloud Data Transfer with Efficient Integrity Auditing for Cloud Computing

Changsong Yang,[1,2] Jun Xiao,[1] Yueling Liu,[3] and Yang Liu[3]

[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]*Guilin University of Electronic Technology, Guilin 541004, China*

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

Received 22 June 2021; Accepted 20 January 2022; Published 23 May 2022

Academic Editor: Jiliang Zhang

Due to the promising market prospects, more and more enterprises invest cloud storage and offer on-demand data storage services, which are characterized by distinct quality; thus, users would dynamically change the cloud service providers and transfer their outsourced data. However, the original cloud server might not honestly transfer the outsourced data for saving overhead, and the outsourced data might be polluted during the transfer process. Therefore, how to achieve secure outsourced data transfer has become a primary concern of users. To solve this issue, we put forward a solution for the issue of provable cloud data transfer, which can also simultaneously realize efficient data integrity auditing. By taking the advantages of Merkle sum hash tree (MSHT), our presented scheme can satisfy verifiability without dependency on a third party auditor (TPA). Meanwhile, the formal security proof can demonstrate that our presented scheme meets all of the expected security requirements. Finally, we implement our presented scheme and offer the precise performance evaluation, which can intuitively show the high-efficiency and practicality of our presented scheme in the real-world applications.

1. Introduction

Cloud storage, a very attractive service offered by cloud service providers, can offer convenient and on-demand data storage services to tenants [1, 2]. By utilizing cloud storage services, the users whose resources are limited can upload their large-scale data to the cloud servers, thus greatly saving their local storage capacity and computation resources [3, 4]. As a result, a great number of individuals and business organizations prefer to embrace cloud storage services. A report of Cisco shows that the number of Internet users reached about 3.6 billion by the end of 2019, and about 55% (near 2 billion) employed cloud storage services. Moreover, the global cloud storage economic output is forecasted to reach 137 billion by the end of 2025 [5].

Because of the promising market prospects, an increasing number of enterprises invests cloud storage and subsequently offers on-demand and convenient data storage services to tenants through the mean of pay-as-you-go [6, 7]. However, their data storage services are equipped with different storage capacities, security, reliability, using price, access speed, and so on. To enjoy more suitable data storage services, users would change the cloud service providers and transfer their outsourced data from one cloud server to another [8]. As a consequence, data transfer becomes more and more frequent in recent years. A report of Cisco shows that the cloud data flow is predicted to occupy 95% of the total data flow by the end of 2021. At the same time, nearly 14% of the cloud data flow would be the data flow among different cloud servers [9].

Although a lot of schemes have been presented for achieving data transfer between two different cloud servers, there are some inherent deficiencies in the existing solutions, resulting in some severe challenges that had not been solidly solved. Firstly, almost all of the previous schemes require a third party auditor (TPA) to manage the related keys or audit the data integrity/transfer results. However, like other ordinary entities, the TPA would refuse to serve because of the continuously increasing computational overhead, resulting in service interruption. Meanwhile, the TPA might be frequently attacked by the adversary, resulting in privacy leakage. Therefore, the TPA would become a technology bottleneck which greatly impedes the rapid popularization of cloud data transfer systems [10].

Secondly, the existing data transfer schemes are not efficient because they contain some complex protocols or calculations, for instance, provable data possession (PDP) protocol, bilinear pairings calculation, and modular exponentiation computation, which will cost plenty of computational resources [11]. To the best of our knowledge, there does not exist such a solution which can simultaneously achieve secure cloud data transfer and efficient integrity auditing without requiring a TPA. Therefore, one of the most important goals in this field is to present a provable cloud data transfer scheme supporting efficient integrity auditing without a TPA. Merkle sum hash tree (MSHT) can be seen as a specific binary tree, which would offer a potential solution to achieve the above goal.

1.1. Our Contributions. We aim to study a challenging issue, i.e., provable cloud data transfer with efficient data integrity auditing in cloud computing environment. Then, we put forward a new scheme, which can achieve provable cloud data transfer and efficient integrity auditing simultaneously. Therefore, the main contributions of this paper are as follows:

We put forward a solution for the issue of provable cloud data transfer with efficient integrity auditing. Specifically, users can dynamically change the cloud service providers and transfer their outsourced data from one cloud server to another securely. Subsequently, users have the ability to efficiently audit the transferred data integrity on target cloud to confirm whether the data blocks are intact.

By taking the advantages of MSHT, our presented solution can satisfy verifiability without dependency on a TPA. In the meantime, we provide the security analysis to formally prove that our presented solution can meet the desired security requirements. Finally, we implement our presented scheme and offer the performance evaluation, which can obviously show the practicality and high-efficiency of our presented solution.

1.2. Related Works. Cloud data integrity auditing has been studied for several tens of years, and there are lots of solutions [12–15]. Ateniese et al. [16] presented the first PDP model in 2007. In their model, sampling was used to achieve data integrity auditing, which is characterized by low computational overhead and communication burden. Then, Erway et al. [17] put forward a model for dynamic PDP, in which they can simultaneously support provable data updates and integrity verification. In 2019, He et al. [18] presented the first public shared data integrity verification protocol, which can support fully dynamic operations with constant storage overhead for verifiers. Huang [19] studied the proxy provable data possession (PPDP) verification model, in which a proxy performed the data possession verification operation for the client. Meanwhile, he adopted bilinear pairing to design an efficient PPDP scheme. Bian and Chang [20] designed a certificateless PDP protocol for multicopy outsourced data stored in different cloud servers. Their novel scheme can resist the key-escrow attack. Chen et al. [21] adopted blockchain to design a DPDP model for smart cities. Then, they used multireplica storage tricks to propose a concrete construction of decentralized PDP.

Except PDP, some other authentication data structures are also utilized to achieve data integrity auditing. Zhang and Dong [22] designed an efficient ID-based outsourced data integrity verification scheme. They proved that their solution was provable secure in the random oracle model. In the meantime, they extended their scheme to achieve batch verification in multiuser scenario. Li et al. [23] suggested to achieve data integrity auditing through fuzzy identity and formalized the corresponding system model. Subsequently, they designed a concrete solution based on fuzzy identity, which is characterized by error tolerance. Yu et al. [24] designed an attribute-based outsourced data integrity verification solution, which achieved attribute privacy-preserving and collusion-resistance. In their scheme, they introduced attribute-based outsourced data integrity verification to solve complex key management challenge. Recently, Fan et al. [25] proposed a secure identity-based aggregate signatures protocol to ensure the integrity of cloud data.

Yue et al. [26] studied the data integrity checking in P2P cloud storage scenarios. Then, they designed a general framework for data integrity checking and adopted blockchain to present a concrete solution. Li and Luo [27] adopted homomorphic encryption and homomorphic signature to establish an integrity auditing scheme for smart grid data, which can prevent the data from accidental errors. Thangavel and Varalakshmi [28] utilized ternary hash tree (THT) to present a data integrity verification scheme, which can support Block-level, File-level, and Replica-level auditing. Meanwhile, their scheme can support data dynamic updates. Zhou et al. [29] studied the problem of data integrity verification in cloud-based electronic medical storage scenarios. Subsequently, the multicopy Merkle hash tree was adopted to present a concrete data integrity auditing scheme. Although the above solutions can support cloud data integrity verification, they cannot support data transfer between two different cloud servers.

Yu et al. [30] presented the first method to simultaneously support outsourced data integrity checking and secure transfer. Wang et al. [31] adopted PDP and homomorphic technology to complete cloud data transfer and integrity auditing. Yang et al. [10] utilized blockchain to present an outsourced data integrity auditing protocol supporting reliable data transfer, in which the data blocks are stored in the consortium blockchain. Xue et al. [32] utilized PDP and rank-based Merkle hash tree (RMHT) to achieve cloud data transfer and integrity checking. Nevertheless, Liu et al. [33] found that there was a serious security flaw in Xue et al.' solution [32]. That is, the dishonest cloud server can successfully forge a random block and generate a related block tag which can successfully pass the auditing. After that, they proposed an improved scheme to overcome this problem. Meanwhile, their solution can reduce the computational overhead in data integrity checking phase.

Wang et al. [34] put forward the notion of PDP with outsourced data transfer (DT-PDP). Subsequently, they utilized bilinear pairing to design a detail DT-PDP scheme. Yang et al. [35] utilized counting Bloom filter (CBF) and hash function to deal with the challenge of cloud data transfer with integrity verification. Ni et al. [36] designed a scheme to move outsourced data securely, in which they utilized PDP to achieve cloud data integrity auditing. Yang et al. [37] made use of vector commitment (VC) to present a provable data migration scheme for cloud storage. They achieved cloud data integrity auditing by utilizing the position blinding of VC. Although the above solutions can realize data transfer and integrity auditing, they either rely on a TPA or contain some complex calculations/protocols.

1.3. Organization. Next, we introduce the organization of the rest of this paper. In Section 2, we describe some preliminaries which are the building blocks of our presented scheme. Section 3 describes the system framework, security challenges, and security goals. Then, the concrete scheme is presented in Section 4. The scheme analysis and performance comparison are, respectively, offered in Section 5 and Section 6. Finally, this paper is simply concluded in Section 7.

2. Merkle Sum Hash Tree

The primitive of Merkle sum hash tree (MSHT) was first put forward by Miao et al. [38]. Generally speaking, MSHT can be seen as an extension of the traditional Merkle hash tree (MHT). Between the MSHT and the MHT, there exist two main differences. On the one hand, every leaf node of MHT merely maintains a data item. However, MSHT can store plenty of data items in each leaf node, as demonstrated in Figure 1. As a result, the height of MSHT will not infinitely increase with the total number of data items. Further, MSHT is able to deal with the challenge that the growth of data items leads to the fast increase in the height in MHT. On the other hand, when computing the hash values in the leaf nodes of MSHT, the data items and the rank are taken as inputs; here, the rank is the number of data items this leaf node maintains. In each nonleaf node, the inputs consist of the concatenation of its left child and right child and the rank of this nonleaf node [39]. At last, a

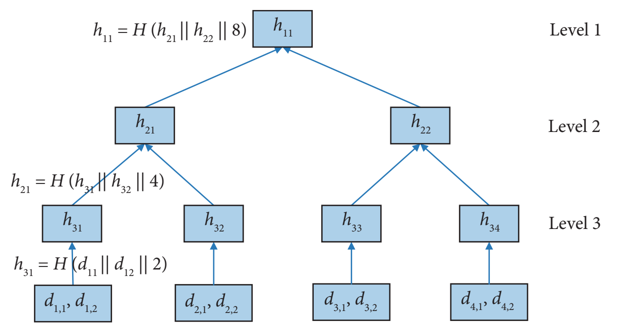

FIGURE 1: An instance in MSHT.

digital signature on the Merkle root is generated by a public key signature scheme.

Like MHT, MSHT is also able to verify data integrity with low computational overhead and communication cost. Specifically, the verifier utilizes the data items which are maintained by a leaf node and its related auxiliary authentication information (AAI) to rebuild the Merkle root h'_{11}. Here, the AAI contains all the hash values of the sibling nodes on the path from the authenticated leaf to the Merkle root. Then, the user compares h_{11} with h'_{11} and verifies the signature. If and only if the signature is valid and the equation $h_{11} = h'_{11}$ holds, the verifier thinks that the data blocks are intact.

3. Problem Statement

In this section, we first establish the system framework. Subsequently, we introduce the security threats and finally identify the security requirements.

3.1. System Framework. In this paper, we study a specific scenario: users change the cloud service providers and move their cloud data from one cloud server to another. Subsequently, the target cloud server audits the transferred data integrity before accepting them. Hence, the system framework of provable cloud data transfer with efficient integrity auditing consists of three entities: a user, an original cloud server, and a target cloud server, as shown in Figure 2.

3.1.1. User. In our system framework, the user owns limited resources, thus cannot store large-scale data in local. Hence, to greatly reduce local cost, the user might want to upload his large-scale data to the cloud server. Later, the user would change the cloud service providers for some subjective or objective factors and move the outsourced data to the target cloud.

3.1.2. Original Cloud Server. The original cloud server (called cloud *A* for short) is an entity that owns almost limitless resources. Hence, the cloud A can provide user with on-demand data storage services, thus maintaining large-scale data for the user. Later, the cloud A transfers the related data according to the user's commands.

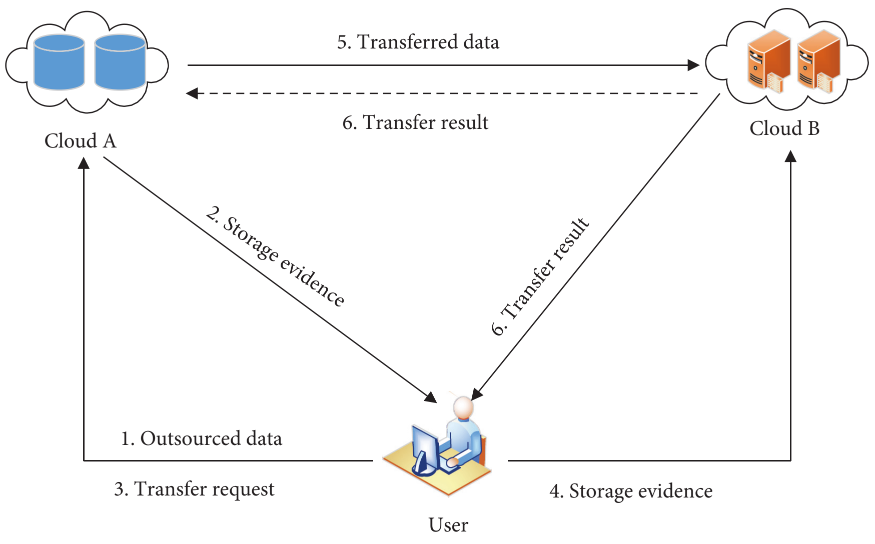

FIGURE 2: The system framework user.

3.1.3. Target Cloud Server. The target cloud server (called cloud *B* for short) connects a lot of distributed storage mediums, network bandwidths, and computation resources to provide user with on-demand data storage service. Specifically, the cloud B receives the transferred data blocks and verifies the data integrity before accepting and maintaining them for the user. Meanwhile, the cloud *B* returns the data transfer result to the user.

In the system framework, we assume that the two cloud servers will never collude together to cheat the user. This assumption is reasonable because of two reasons as follows. Firstly, if the two clouds collude together, they can be seen as an entity and the data transfer between them is meaningless. Secondly, the two clouds belong to two different enterprises, and they are competitors. For economic interests, they will not be on the same side to cheat the user. In addition, we assume that the cloud *B* would never deliberately interrupt the data transfer and maliciously slander the cloud *A*. This assumption is rational since the cloud B aims to offer data storage services to the user. This means that the cloud *B* tries to favourably complete the data transfer, rather than maliciously destroying the transferred data to slander the cloud *A*. Meanwhile, the cloud *B* is not able to obtain any economics interests from slandering the cloud *A* since the user has already terminated the cooperation with the cloud A and stopped its data storage services.

3.2. Security Challenges

3.2. Security Challenges. In our presented scheme, we must seriously consider four security challenges as follows.

3.2.1. Privacy Leakage. The data might contain some privacy information. Then, both the internal attacker and the external attacker might try their best to dig some valuable information from the cloud data. Meanwhile, the hardware failure and software breakdown may also lead to privacy leakage. Therefore, privacy leakage is a serious security challenge that should be considered.

3.2.2. Data Pollution. The selfish cloud administrator might delete some data blocks which are accessed rarely, resulting in data pollution. Meanwhile, the data blocks would be destroyed in the data transfer process. However, the user cannot easily detect the data pollution. Hence, data pollution is another security threat that should be considered.

3.2.3. Dishonest Data Transfer. To achieve cloud data transfer, the cloud A needs to cost plenty of network bandwidths and computational resources, thus greatly reducing the access speed of other users. To guarantee service quality, the cloud A may not sincerely complete the cloud data transfer operation according to the user's command.

3.2.4. Malicious Slander. Both the cloud A and the user will not fully trust each other. Specifically, the cloud A does not execute data transfer operation and slanders that the user has not asked to migrate the data. Moreover, the user has not asked to migrate the data and slanders that the cloud A does not sincerely complete the data transfer operation.

3.3. Security Goals. Our presented scheme must achieve the following expected security goals.

3.3.1. Data Confidentiality. The privacy information should be confidential. As a consequence, the file has to be encrypted before outsourcing to the cloud server. If the related data decryption key is not obtained, the adversary cannot correctly decrypt the related ciphertext.

3.3.2. Data Integrity Auditing. It is ensured that the cloud server honestly maintains the outsourced data. If the cloud server arbitrarily destroys the data, the user has the ability to detect the malicious data pollution through efficiently checking the cloud data integrity.

3.3.3. Provable Data Transfer. It is ensured that all the data blocks would be migrated integrally. If the data blocks have been polluted, the cloud B can detect the data destroy and would refuse to receive the data blocks. Meanwhile, the cloud B will inform the failure of data transfer to the user.

3.3.4. Nonframeability. The fairness is guaranteed in the process of data transfer. That is, the honest participant cannot be slandered and the malicious participant cannot be shielded. If the malicious participant slanders the honest participant, the honest one can prove the sinlessness by himself.

4. Our Presented Scheme

Next, we would present our new scheme in detail. Generally speaking, the user must register and pass the auditing before employing cloud storage service. For simplicity, we let the user have passed the identity auditing and become an authorized customer of the cloud A and the cloud B.

4.1. System Setup. This phase mainly aims to generate some public parameters and related keys. Specifically, the user, the cloud A, and the cloud B generate their public/private key pair (SK_U, PK_U), (SK_A, PK_A), and (SK_B, PK_B), respectively. Then, they, respectively, publish their public key. Meanwhile, the user names the outsourced file F with the file name n_f, which is unique in the cloud storage system.

4.1.1. Data Encryption. For greatly reducing storage burden and computational overhead, the user outsources his large-scale file to the cloud A.

For preventing data privacy from leakage, the user encrypts the file before outsourcing to the cloud A. That is, the user firstly computes a data encryption key $k = H_1$ $(n_f || SK_U)$, where $H_1(\cdot)$ is a secure hash function. Then, the user utilizes key k to efficiently encrypt the file F, *i.e.*, $C = Enc_k$ (F), where Enc is a symmetric encryption algorithm which is IND-CPA secure.

Meanwhile, the user divides the ciphertext C into n' subfiles. Subsequently, the user might insert $n - n'$ random subfiles into these n' subfiles at random positions. Next, the user further divides every subfile into s data blocks. Therefore, the outsourced file can be denoted as $f = (f_1, f_2, \ldots, f_n) = \{f_{i,j}\}_{1\leq i\leq n, 1\leq j\leq s}$. Finally, the user sends the large-scale data set f to the cloud A.

4.1.2. Data Outsourcing. The cloud A stores the data and generates a proof, which can be utilized to audit the data storage consequence by the user.

The cloud A builds a MSHT to store the outsourced file f. More specific, every leaf node maintains a subfile, i.e., leaf node Ni maintains subfile f_i. Then, the cloud A utilizes his private SK_A to generate a signature sig R on the Merkle root H_R. That is, $\mathrm{sig}_R = \mathrm{Sign}_{\mathrm{SK_A}}$ (H_R), where Sign is the signature generation algorithm of ECDSA. Finally, the cloud A returns H_R and sig_R to the user.

Upon receiving of H_R and sig_R, the user verifies the data storage consequence by auditing the storage proof. The user firstly generates a set of leaf node indices λ that contains $n/2$ numbers which are randomly chosen from $1, 2, \ldots, n$. Then, the user retrieves the related AAI set $\{\Phi_q\}_{q\in\lambda}$. Next, the user utilizes the data blocks which are stored in $\{N_q\}_{q\in\lambda}$ and $\{\Phi_q\}_{q\in\lambda}$ to rebuild a Merkle root H_R'. Finally, the user compares H_R with H_R' and audits sig_{R}. If and only if equation $H_R = H_R'$ holds and sig_{R} is a valid signature on H_R', the user believes that the outsourced file f has been honestly maintained. Finally, the user would delete the local copy of f.

4.1.3. Data Transfer. The user changes the cloud service providers and then transfers his data to the cloud B for more suitable data storage services.

The user first generates a set of subfile indices φ that identifies the subfiles which are needed to be transferred to the cloud B. Then, the user generates a signature $\mathrm{sig}_t = \mathrm{Sign}_{\mathrm{SK_U}}$ (transfer $||n||\varphi||T_t$), where T_t represents the timestamping. Further, the user can generate a data transfer command $TQ = (nf, \varphi, T_t, sig_t)$. Finally, the user sends the transfer command TQ to the cloud A and the cloud B. Meanwhile, the user sends the signature sig R and the Merkle root H_R to the cloud B.

Once received TQ, the cloud A checks whether $_{TQ}$ is valid through signature auditing. That is, the cloud A audits if the signature sig_t is valid or not. If sig_t is invalid, the cloud A terminates and outputs failure; otherwise, the cloud A migrates the subfiles $\{f_i\}_{i\in\varphi}$ to the cloud B, along with the AAI set $\Phi_{i\in\varphi}\{i\}$. At the same time, the cloud A generates a signature $(TA)_A = \mathrm{Sign}_{SK}(^{TQ})$ and returns it to the user.

On receipt of subfiles $\{f_i\}_{i\in\varphi}$, the cloud B first audits the integrity of data blocks before accepting them. That is, the cloud B utilizes the received subfiles $\{f_i\}_{i\in\varphi}$ and the AAI set $\{\Phi_i\}_{i\in\varphi}$ to rebuild the MSHT and obtain a new Merkle root H_R^*. Then, the cloud B compares H_R^* with H_R and verifies the signature sig_R. If $H_R = H_R^*$ holds and signature sig_R is valid, the cloud B thinks that the subfiles $\{f_i\}_{i\in\varphi}$ are integrated. Finally, the cloud B informs the user the success of data transfer.

5. Scheme Analysis

We firstly analyse the security of our presented scheme. Subsequently, we offer the theoretical computational complexity analysis and approximate efficiency comparison through numeric analysis.

5.1. Security Proof. We provide the formal security proof to show that our presented scheme can meet all of the expected security requirements.

Theorem 1. *Our scheme can achieve data confidentiality.*

Proof. In our presented scheme, the cloud data are encrypted by the user through a symmetric encryption algorithm before uploading. The security of the cloud data is ensured by the chosen encryption algorithm. Hence, if do not get the related decryption key, the ciphertext is IND-CPA secure. The detailed security proof process is similar to scheme [40], so we ignore it here. Hence, if an attacker does not own the related decryption key, he cannot correctly decrypt the ciphertext. That is, our presented scheme is able to guarantee that no other than the user can correctly decrypt the ciphertext. Therefore, the cloud data confidentiality can be guaranteed. □

Theorem 2. *Our scheme can achieve data integrity auditing.*

Proof. In order to realize data integrity verification, the user should rebuild the MSHT and verify the signature. Specifically, assume that the user prefers to audit the integrity of data blocks which are stored in the leaf node N_k, the user first receives these data blocks (i.e., cloud data blocks $f_{k,1}, f_{k,2}, \ldots, f_{k,s}$, where $1 \le k \le n$). Meanwhile, the user can obtain the related auxiliary authentication information Φ_k. Then, the user utilizes $f_{k,1}, f_{k,2}, \ldots, f_{k,s}$ and Φ_k to rebuild the Merkle root H_r and compares it with H_R. In the meantime, the user audits the validity of the signature sig_R. If data blocks $f_{k,1}, f_{k,2}, \ldots, f_{k,s}$ are maliciously polluted, the cloud server cannot successfully forge new data blocks $f'_{k,1}, f'_{k,2}, \ldots, f'_{k,s}$ to let the equation $H_r = H_R$ hold. Therefore, when $H_r = H_R$ holds and sig_R is valid, the user thinks that the data blocks $f_{k,1}, f_{k,2}, \ldots, f_{k,s}$ are intact. That is to say, our presented scheme can achieve cloud data integrity auditing. □

Theorem 3. *Our scheme can achieve provable data transfer.*

Proof. In our presented scheme, the user sends the cloud data transfer command to the two clouds (i.e., both cloud A and cloud B). Hence, if the cloud A arbitrarily transfers the cloud data without the user's command, the cloud B can detect the arbitrary data transfer. Further, the cloud B refuses to accept the transferred cloud data and informs the malicious data transfer to the user. Conversely, if the user indeed prefers to migrate the cloud data, the cloud A transfers the data to the cloud B. Then, the cloud B would audit the integrity of the received data blocks before accepting them. If and only if the cloud data blocks are not destroyed, the cloud B would accept them and inform the data transfer result to the user. Note that the cloud B tries its best to favourably complete the data transfer, rather than destroying the transferred data to slander the cloud A maliciously. Meanwhile, the cloud B would never collude with the cloud A to cheat the user. Hence, the data transfer result returned by the cloud B is trusted. Further, our presented scheme can achieve provable cloud data transfer. □

Theorem 4. *Our scheme can achieve nonframeability.*

Proof. We analyse the nonframeability from the following two aspects.

Dishonest User. If the user is dishonest, he would slander the cloud A maliciously. Specifically, the user has asked the cloud A to migrate the outsourced data blocks. Nevertheless, the user would claim that the cloud A maliciously interrupted the storage services and arbitrarily migrated the related data blocks to the cloud B. In this scenario, the cloud A can show the data transfer command $TQ = (n_f, \varphi, T_t, sig_t)$ sent by the user.

Note that the data transfer command TQ contains a signature sig_t. At the same, except the user, nobody can compute it without private key SK_U. In other words, the data transfer command TQ can only be generated by the user and nobody else can forge it successfully.

Therefore, the data transfer command TQ could be seen as a proof which can demonstrate that the user has asked the cloud A to migrate the data blocks to the cloud B. Further, the cloud A can prove his sinlessness by demonstrating the data transfer command TQ. As a result, the user cannot successfully slander the cloud A.

Malicious Cloud A. Similarly, the cloud A would also slander the user maliciously. Specifically, the user has asked to transfer the data blocks to the cloud B; however, the cloud A did not sincerely carry out the data transfer command. When the dishonest acts of cloud A are detected by the user, the cloud A would slander that the user did not require to migrate the cloud data. In this setting, the user is able to demonstrate the signature sig_A which is sent from the cloud A. Note that the signature sig_A is the response to the data transfer command of the user. Meanwhile, except the cloud A, nobody can compute the signature sig_A without private key SK_A. Therefore, sig_A can be seen as a proof to demonstrate that the cloud A has received the data transfer command TQ from the user and has responded to it. Further, the malicious cloud A cannot successfully deny the data transfer command and maliciously slander the user. □

5.2. Numeric Analysis and Efficiency Comparison. Next, we compare the theoretical computational complexity of our presented scheme with the existing scheme [35], which can achieve similar functionalities. For simplicity, we ignore the communication cost and only consider the main computational cost in each phase. Moreover, we first define some symbols which will be utilized in the comparison.

Specifically, we denote by ε the symmetrical encryption calculation and H of an operation for computing a hash value. Meanwhile, we adopt symbols S and V to, respectively, represent the signature generation operation and verification operation. Moreover, symbol n and symbol l are, respectively, used to define the total number of cloud data blocks and the number of transferred cloud data blocks. In our comparison, we assume that each leaf node of MSHT maintains one data block. Subsequently, the comparison is described in Table 1.

From Table 1, we can easily obtain the following two findings. On the one hand, the existing scheme [35] needs to perform more hash calculations than our presented scheme to achieve data encryption. In this phase, both the two schemes

Table 1: Complexity comparison.

Scheme	Scheme [35]	Our presented scheme
Data encryption	$1\varepsilon + (n+1)H$	$1\varepsilon + 1H$
Data outsourcing	$1S + 1V + 30nH$	$1S + 1V + (2n + n/2 \log_2 n)H$
Data transfer	$3S + 3V + 31lH$	$2S + 2V + l\log_2 nH$

need to compute a hash value to generate the data encryption key and then utilize it to encrypt the outsourced file. Nevertheless, the existing scheme [35] needs n more hash calculations to compute the hash value of every data block. On the other hand, our presented scheme costs much less computational overhead than the existing scheme [35] in data outsourcing and data transfer phases. As a result, we think that our presented scheme would be more attractive than solution [35].

6. Performance Evaluation

We implement our presented scheme and offer the performance evaluation. Specifically, all of the simulation experiments are carried out on a Desktop equipped with Linux operation system, Intel(R) Core(TM) i7-7700 processors running at 3.6 GHz and 8 GB RAM. Meanwhile, we implement the related cryptographic algorithms with the open-secure sockets layer library and the pairing-based cryptography library. For simplicity, we ignore the communication overhead and merely measure the time overhead of performing the main operations in every phase.

6.1. Computational Overhead of Data Encryption. The main computations are data encryption key generation and data encryption operation, which are closely related to the size of encrypted file. Therefore, in this simulation experiment, we increase the size of encrypted file from 1 MB to 10 MB with a step for 1 MB. In the meantime, we assume that the total number of data blocks is 20,000. Subsequently, we measure the approximate time overhead, which is presented in Figure 3.

From Figure 3, we are able to readily discover that the overhead approximatively linearly rises with the size of encrypted file. In the meantime, the growth rates of time cost in both the two schemes are almost the same. Nevertheless, the computational overhead of our presented solution is obviously less than that of scheme [35]. For instance, when the size of encrypted file is 5 MB, our presented scheme might cost approximate 12.6 milliseconds, while the scheme [35] would cost more than 13.3 milliseconds. Predictably, in this data encryption phase, scheme [35] will cost more time cost than our presented scheme under the same size of encrypted file. Moreover, note that for every file, this data encryption phase is one-time, and it could be performed offline by the user. Therefore, in this process, we can think that our presented solution is more efficient.

6.2. Computational Overhead of Data Outsourcing. The major computations are hash calculations, signature generation, and auditing calculations. In order to present the comparison more clearly, we can divide this phase into two portions for simplicity, that is, data storage and data storage result verification.

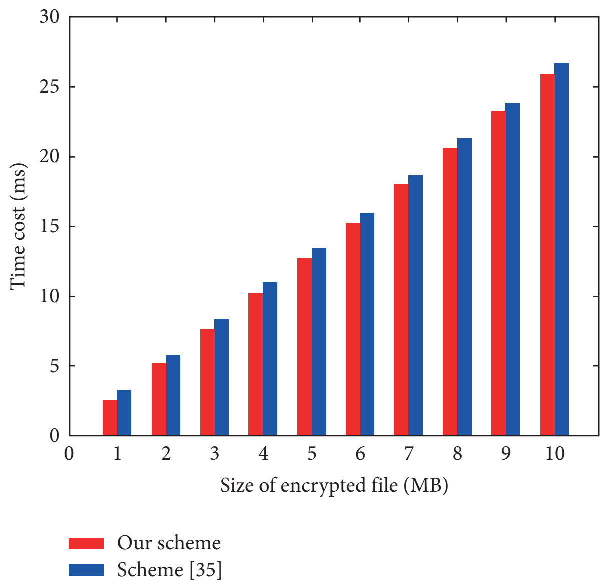

Figure 3: Time cost of data encryption.

In data storage phase, the major time cost is from the calculations for generating the data storage proof, which is closely related to the total number of cloud data blocks. Therefore, in the simulation experiment, we would rise the number of cloud data blocks from 200 to 2,000 with a step for 200. Subsequently, we measure the time cost, which is presented in Figure 4.

We could obviously have two findings from Figure 4. Firstly, we can discover that the overhead of our presented solution is obviously less than that of scheme [35]. For instance, scheme [35] requires more than 13.0 milliseconds when the number of cloud data blocks is 2,000. Nevertheless, our presented scheme requires nearly 1.5 milliseconds. Secondly, we can see that both the time overheads of the two schemes approximatively linearly rise with the total number of cloud data blocks. Nevertheless, the time overhead of our presented scheme grows much slower than that of the previous solution [35]. From the above two findings, we can think that our presented scheme would always cost less time overhead than scheme [35] under the same number of cloud data blocks. As a result, we can think that our presented scheme has higher efficiency than the existing scheme [35] to realize data storage.

Similarly, in the data storage result verification process, the major computational cost is from the calculations for auditing the correctness of data storage proof, which is also quite closely related to the number of cloud data blocks. As a consequence, we would also rise the total number of cloud data blocks from 200 to 2,000 with a step for 200 in the simulation experiment. Subsequently, we measure the approximate time overhead, as demonstrated in Figure 5.

We can directly see that both the time overheads of the two schemes approximatively linearly rise with the total number of cloud data blocks. However, the time overhead of

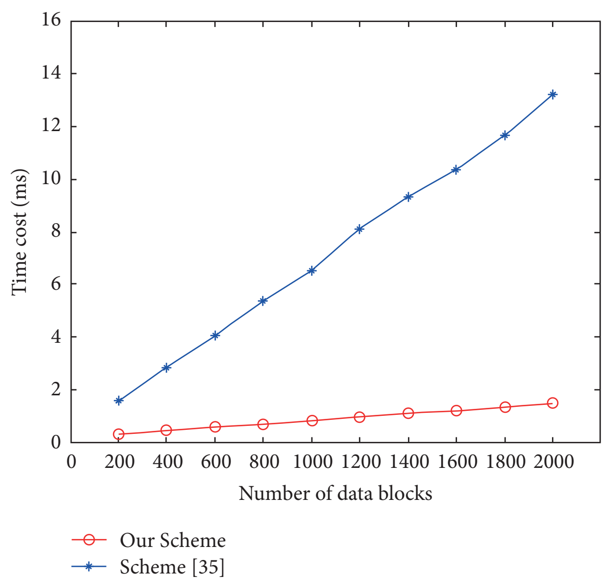

FIGURE 4: Time cost of data storage.

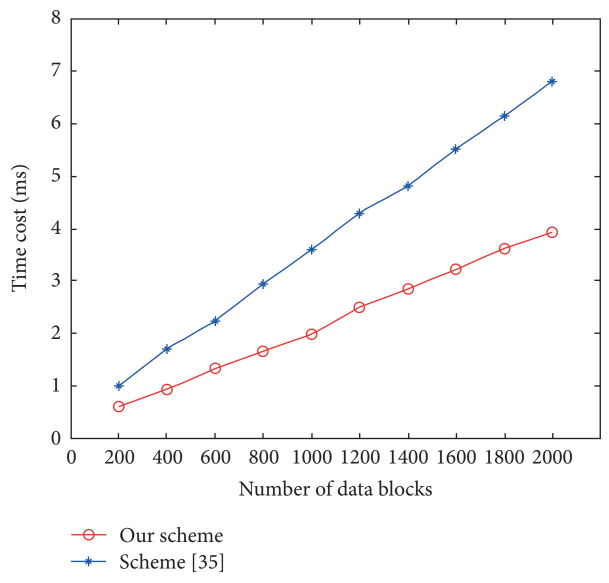

FIGURE 5: Time cost of storage result verification.

scheme [35] grows faster than that of our presented scheme. At the same time, we can intuitively discover that our presented scheme requires much less time overhead than scheme [35]. For instance, scheme [35] requires more than 6.7 milliseconds when the number of cloud data blocks is 2,000. Nevertheless, our presented scheme only requires near 4.0 milliseconds. Therefore, we can forecast that when the number of cloud data blocks is the same, the time overhead of our presented scheme will be less than that of scheme [35]. Moreover, note that this process would be performed by the user, who is resource-constraint. Hence, this process would greatly affect the overall efficiency of the two schemes. As a result, we are able to think that our presented scheme has more advantages in efficiency than scheme [35].

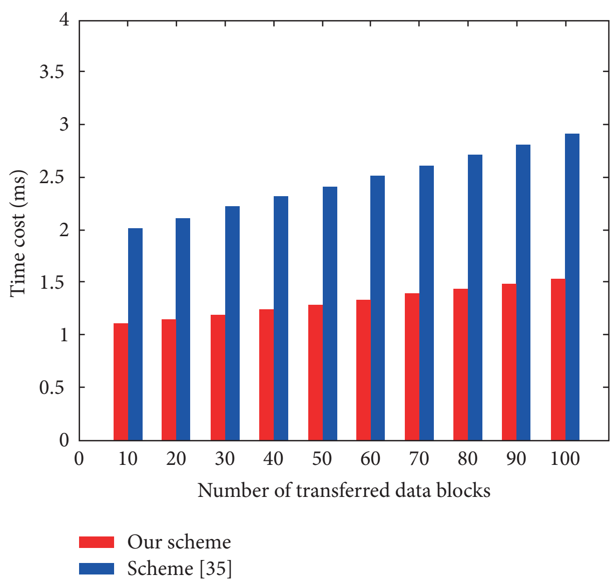

FIGURE 6: Time cost of data transfer.

As a whole, according to the time overhead comparison presented in Figures 4 and 5, we can think that our presented solution is obviously much more efficient than scheme [35] in data outsourcing phase.

6.3. Computational Overhead of Data Transfer. In this simulation experiment, the major time overhead is from the calculations of verifying the data integrity, which is closely related to the number of transferred cloud data blocks. Because of the instability of network, we ignore the communication overhead. Therefore, we rise the number of transferred cloud data blocks from 10 to 100 with a step for 10. Subsequently, we measure the time cost of performing the main operations, which is shown in Figure 6. We intuitively discover that the time overhead of the two schemes both rise with the number of transferred cloud data blocks. In the meantime, the time overhead of scheme [35] grows much faster than that of our presented scheme. In addition, our presented scheme needs much less time overhead than scheme [35]. As a consequence, we are able to easily think that our presented scheme has some obvious advantages over scheme [35] in data transfer phase.

According to the time overhead comparison demonstrated in Figures 3–6, we are able to distinctly find that our presented scheme always costs less time cost than scheme [35] in each phase. Further, we can easily think that the total time overhead of our presented scheme is much less than that of scheme [35]. As a result, our presented scheme not only has more advantages in efficiency of every phase but also has more advantages in overall efficiency.

7. Conclusions and Future Works

In this paper, we studied a severe challenge, that is, provable cloud data transfer with efficient data integrity auditing. To deal with this challenging problem, we adopt MSHT to present a new solution. Our presented scheme enabled the user to transfer the cloud data from one cloud server to another and then check the transferred cloud data integrity on the target cloud server. By taking the advantages of MSHT, our presented scheme does not require to rely on a TPA, thus can avoid the problems of service interruption and privacy leakage caused by the single-point-of-failure of TPA. Moreover, the formal security analysis can prove that our presented solution can meet all of the expected security requirements. Finally, we implement our presented scheme and offer the performance evaluation, which can intuitively prove the high-efficiency and practicality of our presented solution.

Note that all of the existing outsourced data transfer solutions only consider the scenarios that the outsourced data blocks are transferred from one cloud server to another one, which are not suitable for the settings that the outsourced data blocks are transferred to two or more cloud servers. However, the user might want to transfer the outsourced data blocks to more than one cloud servers. Therefore, we will study the problem of secure data transfer under multitarget cloud servers in the future.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Science and Technology Program of Guangxi (No. AD20297028), the Natural Science Foundation of Guangxi (No. 2020GXNSFBA297132), the Innovation Project of GUET Graduate Education (No. 2021YCXS063), the Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS202128), the National Key R&D Program of China (No. 2020YFB1006003), the National Natural Science Foundation of China (No. 61772150), and the Guangdong Key R&D Program (No. 2020B0101090002).

References

[1] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, “Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2018.

[2] 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, no. 3, pp. 608–619, 2018.

[3] M. Ma, M. Luo, S. Fan, and D. Feng, “An efficient pairing-free certificateless searchable public key encryption for cloud-based IIoT,” *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8850520, 2020.

[4] V. Chang and G. Wills, “A model to compare cloud and non-cloud storage of big data,” *Future Generation Computer Systems*, vol. 57, pp. 56–76, 2016.

[5] Cisco global cloud index, “Forecast and methodology,” 2014.

[6] W.-B. Kim, D. Seo, D. Kim, and I.-Y. Lee, “Group delegated ID-based proxy reencryption for the enterprise IoT-cloud storage environment,” *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 7641389, 11 pages, 2021.

[7] 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.

[8] C. Yang, X. Tao, S. Wang, and F. Zhao, “Data integrity checking supporting reliable data migration in cloud storage,” in *Proceedings of the The 15th International Conference on Wireless Algorithms*, pp. 615–626, Qingdao, China, 2020.

[9] Cisco global Cloud index, “Forecast and methodology,” 2014, https://www.cisco.com/c/en/us-/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf.

[10] 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, Article ID 103184, 2021.

[11] Y. Liu, X. A. Wang, Y. Cao, D. Tang, and X. Yang, “Improved provable data transfer from provable data possession and deletion in cloud storage,” in *Proceedings of the International Conference on Intelligent Networking and Collaborative Systems*, pp. 445–452, Bratislava, Slovakia, 2018.

[12] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, “Cooperative provable data possession for integrity verification in multicloud storage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[13] G. Neenu and S. Bawa, “Comparative analysis of cloud data integrity auditing protocols,” *Journal of Network and Computer Applications*, vol. 66, pp. 17–32, 2016.

[14] 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.

[15] J. Tian and J. Xuan, “Cloud data integrity verification scheme for associated tags,” *Computers & Security*, vol. 95, Article ID 101847, 2020.

[16] G. Ateniese, R. Burns, and R. Curtmola, “Provable data possession at untrusted stores,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, Alexandria, VI, USA, 2007.

[17] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, “Dynamic provable data possession,” *ACM Transactions on Information and System Security*, vol. 17, no. 4, pp. 1–29, 2015.

[18] 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, pp. 1394–1408, 2019.

[19] H. Wang, “Proxy provable data possession in public clouds,” *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2012.

[20] G. Bian and J. Chang, “Certificateless provable data possession protocol for the multiple copies and clouds case,” *IEEE Access*, vol. 8, pp. 102958–102970, 2020.

[21] 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.

[22] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Information Sciences*, vol. 343-344, pp. 1–14, 2016.

[23] 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, 2017.

[24] 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, 2017.

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

[26] D. Yue, R. li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in P2P cloud storage," in *Proceedings of the IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 561–568, Singapore, 2018.

[27] F. Li and B. Luo, "Preserving data integrity for smart grid data aggregation," in *Proceedings of the IEEE Third International Conference on smart grid communications (SmartGridComm)*, pp. 366–371, Tainan, Taiwan, 2021.

[28] M. Thangavel and P. Varalakshm, "Enabling ternary hash tree based integrity verification for secure cloud data storage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 12, pp. 2351–2362, 2019.

[29] L. Zhou, A. Fu, Y. Mu, H. Wang, S. Yu, and Y. Sun, "Multicopy provable data possession scheme supporting data dynamics for cloud-based Electronic Medical Record system," *Information Sciences*, vol. 545, pp. 254–276, 2021.

[30] Y. Yu, J. Ni, W. Wu, and Y. Wang, "Provable data possession supporting secure data transfer for cloud storage," in *Proceedings of the 2015 10th International Conference on Broadband and Wireless Computing, Communication and Application*, pp. 38–42, Krakow, Poland, 2015.

[31] Y. Wang, X. Tao, J. Ni, and Y. Yu, "Data integrity checking with reliable data transfer for secure cloud storage," *International Journal of Web and Grid Services*, vol. 14, no. 1, pp. 106–121, 2018.

[32] L. Xue, J. Ni, Y. Li, and J. Shen, "Provable data transfer from provable data possession and deletion in cloud storage," *Computer Standards & Interfaces*, vol. 54, no. 1, pp. 46–54, 2017.

[33] Y. Liu, S. Xiao, H. Wang, and X. Wang, "New provable data transfer from provable data possession and deletion for secure cloud storage," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, Article ID 1550147719842493, 2019.

[34] H. Wang, D. He, A. Fu, Q. Li, and Q. Wang, "Provable data possession with outsourced data transfer," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1929–1939, 2019.

[35] 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.

[36] J. Ni, X. Lin, K. Zhang, Y. Yu, and X. S. Shen, "Secure outsourced data transfer with integrity verification in cloud storage," in *Proceedings of the 2016 IEEE/CIC International Conference on Communications in China*, pp. 1–6, Chengdu, China, 2016.

[37] C. Yang, J. Wang, X. Tao, and X. Chen, "Publicly verifiable data transfer and deletion scheme for cloud storage," in *Proceedings of the 20th International Conference on Information and Communications Security*, pp. 445–458, Lille, France, 2018.

[38] 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, 2017.

[39] 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, Article ID 1550147720958294, 2020.

[40] E. Sergei and O. Günther, "Encryption techniques for secure database outsourcing," in *Proceedings of the 12th European Symposium on Research in Computer Security*, pp. 327–342, Dresden, Germany, 2007.