Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 7609543, 12 pages
https://doi.org/10.1155/2021/7609543

Research Article

A Hierarchical Provable Massive Data Migration Method under Multicloud Storage

Ma Haifeng,[1,2] Yu HaiTao,[3] Zhang Ji,[1] Wang Junhua,[1] Xue Qingshui,[1] and Yang Jiahai[2]

[1]*School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China*
[2]*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China*
[3]*School of Tourism and Landscape, Guilin University of Technology, Guilin 541004, China*

Correspondence should be addressed to Yu HaiTao; albertyht@163.com

Received 27 June 2021; Revised 8 October 2021; Accepted 9 November 2021; Published 17 December 2021

Academic Editor: Ding Wang

Now, many users have stored files on multiple clouds, and sometime, a large number of files are migrated between clouds. Because cloud providers and cloud servers are not entirely trusted, the corruption of user's files event occur from time to time during the processes of storage and migration. Therefore, integrity verification must be performed, and the time verification overhead should be as low as possible. The existing provable data migrate methods still have the issue of high time overhead when a large number of files are migrated. Aiming at this problem, this paper proposed a hierarchical provable data migration method, which can provide the efficiency of data transfer integrity verification when moving large number of continuous files between clouds. In this paper, the proposed method is described in detail as well as the security analysis performance evaluation. The results proved that the proposed method can significantly decrease the detection latency of files transfer between clouds.

1. Introduction

With the booming development of cloud computing, Internet of Things, and mobile devices etc., our lives have changed profoundly due to convenience and challenge to us. Mobile devices play a central role in the emerging Internet of Everything era. Various kinds of mobile devices appear in our daily life, which facilitate a wide variety of online services [1] due to the provision of continuous and reliable connectivity. However, many emerging challenges have to be considered while deploying mobile devices in large scale due to different characteristics from traditional personal computers, servers, and laptops [2]. For example, mobile devices no longer meet the requirements for limited storage capacity, and cloud storage is the ultimate solution. It can provide massive storage and can reduce overhead of data management significantly. As a result, a large number of users already use cloud storage services and some of them have saved files on multiple clouds. Besides access data, cloud storage users sometimes migrate massive files between clouds. However, because the cloud service providers and cloud servers are not completely trusted, files corruption on the cloud occurs frequently during migration process. Therefore, data integrity verification must be performed during the process of cloud storage and files migration. At the same time, the original data was confirmed to delete, and it also requires verification overhead.

Data integrity verification of cloud storage is to verify whether the user's data on cloud storage servers are in good condition so that it is avoided that data of users on cloud storage are tampered with or removed. At present, researches on cloud storage integrity mainly focus on two aspects: provable data possession (PDP) [3–6] and proof of retrievability (POR) [7, 8]. Based on pseudorandom sampling, their basic idea is to decrease communication overhead by taking advantage of some form of challenge-response protocol and probabilistic inspection method. PDP proves that files of users are integrated by means of challenge-response protocol. Although PDP can detect higher than a certain percentage of data corruption, it is guaranteed that files are retrievable. Similar

to PDP, POR also uses challenge-response protocols to prove the integrity of files. In addition, users are capable of retrieving files from servers with high probability.

Although many verification methods of data migration between clouds are proposed currently, most of these solutions need to verify all the files. Because complex cipher operations are necessary when authentication is implemented, the authentication of massive files will result in very high computation and communication overhead and even authentication failure. Motivated by the problem, this paper proposed a hierarchical provable data migration method for the scenario of massive data migration between clouds, which is a kind of efficient selection authentication method and can significantly reduce computation and communication overheads as well as bandwidth requirements.

Our contribution can be summarized as follows:

(1) Pinpointing the source of the efficiency problem in Xue et al.'s scheme [9]. This paper proposes an efficient and secure data migration verification method. It provides efficient integrity protection with strong evidence that untrustworthy server cannot pass the verification unless it indeed keeps the data intact. The authentication time of massive data migration between clouds can be obviously reduced, thus improving the authentication efficiency of file migration

(2) This paper give a security analysis of the proposed scheme based on our security model and prove that our scheme is secure against internal and external attacks. Moreover, this paper evaluate the performance of proposed scheme and put comparisons with Xue et al.'s scheme

(3) The proposed method is a verification mode, and it can be combined with other provable data possession and provable data transfer method to further improve authentication efficiency. It also can be applied to many data integration authentication scenarios, such as single cloud storage, multicloud storage, and Internet of things

(4) The security intensity of the proposed method can be adjusted, either high security intensity and high overhead with fine-grained authentication, or low security intensity and low overhead with coarse-grained authentication

2. Related Work

Ateniese et al. [3] proposed PDP model, a light weight method for remote data authentication. The disadvantages of PDP lie in: the times of data update and authentication are limited and it does not support data with dynamic types. For the reason, an improved PDP [10] is proposed based on public key encryption support files. Different from statistic PDP, the improved PDP can implement operations of update and delete for file blocks. However, the improved PDP does not still support insert operation of file blocks.

Erway et al. proposed DPDP [11], a framework for dynamic provable data possession. On the basis of PDP, DPDP provides the operations of insert, update, and delete for file blocks by means of an authenticated skip list. Wang et al. presented another improved PDP scheme supporting full dynamic operation. The scheme guarantees the correctness of the data block in the position using Merkle hash tree and guarantees the integrity of data value using BLS signature. Curtmola [12] proposed MR-PDP scheme for the verification of multiple different replicas. Etemad [13] also proposed a scheme which can verify not only the contents of files but also the number of replicas. However, due to more encryption computing caused by multiple replicas operation, additional computational overhead is increased.

Xue et al. [9] proposed a provable data transmission scheme, the data owner can migrate the data from one cloud server to another one and check the data integrity through provable data possession scheme, it allows a semitrusted cloud server to generate a simple proof to prove that the data deletion command was executed correctly, and the transmitted data was deleted correctly. Liu et al. [14] designed an improved new provable data transfer scheme, which can resist more attack and more efficient in data integrity checking compared with scheme [9]. Wang et al. [15] proposes an auditing scheme for cloud storage services, and the scheme has the properties of secure data transfer, provable data erasure, high error detection probability, and confidential data storage. The above three schemes are all efficient data transfer authentication methods. However, in above three schemes, all the transfer files between clouds need to be authenticated.

Wang et al. proposed the provable data possession with outsourced data transfer (DT-PDP) [16] scheme. It can satisfy the following: the purchased data integrity and privacy can be ensured; the data transferability's computation can be outsourced to the public cloud servers. Reference [17] proposed a cryptographic-accumulator provable data possession (CAPDP) method, which is based on the RSA password accumulator to verify the integrity of the outsourced data, reducing the data owners' burden and overhead of the verification process. Reference [18] proposed the Tagging of Outsourced Data (TOD), where a tag is used to generate and verify files. Users with lower overhead can achieve data verifiability of public and private and can resist label forgery and tampering. However, because TOD is designed based on the conventional sampling inspection, it is not guaranteed that TOD can detect cloud service provider's illegal behaviors with high probability.

Juel et al. proposed POR model [7] which can guarantee the possession and retrievability of data files on remote servers by means of spot-checking and error-correcting codes, respectively. "sentinels," some special blocks for detections, are embedded into data files at random. The disadvantages of POR lie in queries that are performed at fixed times at clients, and public verifiability is not supported. Combining with the research work of Juel and Shacham, Bowers et al. provided an improved version of POR protocol [8].

Shacham et al. [19] used Ateniese's homomorphism authentication tag to construct a homomorphism authenticator

based on BLS signatures. Because short signatures contribute to the aggregation of individual signatures, a very small authenticated value is necessary for public verifiability. The proposed scheme not only decreases the communication overhead for verifications but also supports challenges with unlimited times.

Wang [20] proposed a scheme where file data privacy is guaranteed by the introduction of a random number in basic BLS signature scheme during challenge-response processes. The disadvantage of the scheme is not supporting insert operation. Reference [21] considers preventing the indistinguishability and privacy of the auditor in the outsourcing data integrity audit. Reference [22] proposed a security audit scheme based on identity protection and a multireplica data scheme.

Reference [23] proposed a conditional identity privacy-preserving mechanism for cloud-based WBANs (wireless body area networks). This scheme is mainly used to protect the identity privacy and sensitive information of patients's EHRs. They used public auditing to ensure that the data integrity of patients and prevents malicious cloud service providers from returning error audit reports. Reference [24] proposed a forward secure PEKS scheme (FS-PEKS) based on lattice assumptions for cloud-assisted Industrial Internet of Things (IIoT). They integrate a lattice-based delegation mechanism with keyword search into FS-PEKS to achieve forward security, and the security of the system is still guaranteed when the keys are compromised by the adversaries. Reference [25] proposed a privacy-preserving anonymous authentication scheme for WBANs. The scheme can provide the message integrity and develop a conditional tracking system to track the misbehaving doctors in the WBAN

3. Preliminaries

3.1. Bilinear Mapping. Assume that (G, G^T) is a cyclic group of the same prime order p, g is the generator of G, and $\widehat{e} : G \times G = G_T$ is a bilinear map if the following properties are satisfied:

(1) Bilinear: for all $x, y \in Z_p$, $\widehat{e}(g^x, g^y) = \widehat{e}(g, g)^{xy}$

(2) Nondegeneracy: $\widehat{e}(g, g) \neq 1_{G_T}$, the identity element is in G_T

(3) Efficient computability: for all $x, y \in Z_p$, $\widehat{e}(g^x, g^y)$ is efficient and computable

3.2. RMHT. Merkle Tree, also known as Merkle Hash Tree (MHT), is a classical data integrity verification structure, which can effectively verify whether an element has been tampered with [26]. When a Merkle tree is built on a data set S, the hash value of each element on S is taken as the leaf node of the tree, and each inner node is the hash value of its left and right children [27].

In Reference [9], an extended Hash Tree Rank-based Merkle Hash Tree (RMHT) was proposed, which is similar to MHT. The difference is that the input of an internal node is not only the left node and the right node but also the Rank. The Rank refers to the number of leaves of the node, as shown in Figure 1. Nodes $h_1 - h_8$ have one leaf node, and their level is 1; h_c, h_d, h_e, and h_f have two leaf nodes, and their level is 2, but root node h_r has eight leaves and its level is 8.

4. Data Migration Model and Authentication Framework

4.1. Data Migration Model. Provable data transfer model includes four entities: Data Owner (DO), Third Proxy Agent (TPA), and two clouds [9, 28]. The data migration model is shown in Figure 2. The clouds have a large amount of storage resources and strong computing power. The data owner, with limited resources, may be PC or smart mobile devices. They are generally limited in computation capability and restricted storage and energy and use the services provided by CSP [29]. TPA has capabilities that the data owner does not have, and it can initiate authentication requests on behalf of the data owner [30, 31].

In Figure 2, DO chooses cloud A to store the data, and cloud A is used to periodically detect the data integrity. If the user wants to change CSP, he or she first selects cloud B to store data and send data transfer request to cloud A. Cloud A sends the corresponding data to cloud B, and cloud B deletes the migrated data.

In order to ensure the data integrity of cloud B and the secure deletion of data on cloud A, the user sends a verification request to TPA, and TPA verifies whether the data on cloud B is integrity and the data on cloud A has been deleted. And then, TPA returns the verification results to the user. Finally, the user can still request TPA to continue periodically detect the remaining files on cloud A and new files stored on cloud B.

4.2. Cloud Data Migration Integrity Verification Framework. Based on reference [9], the design framework of this method extends PDP model supporting provable data transmission, including five stages: KeyGen, Store, Transfer, DeletCheck, and IntegCheck [30].

(1) KeyGen: the probabilistic algorithm is run by DO to produce private key and public key, and public key is authenticated by Certification Authority (CA)

(2) Store: It is an outsourced data generation algorithm. DO generates files and corresponding tags. The files are divided into several blocks, a random number of probe blocks are inserted into the files, and the owner generates a polynomial tag for each block. Then, DO builds the RMHT (rank-based Merkle Hash Tree) and signs the Root. RMHT is similar to Merkle Hash Tree. The difference is that the leaf nodes of RMHT are not only the connection value of the left and right nodes but also contain the level of the current node

(3) Transfer: a secure data transmission algorithm is run by DO and clouds A and B. As DO transfer data from cloud A to cloud B, the data integrity on cloud

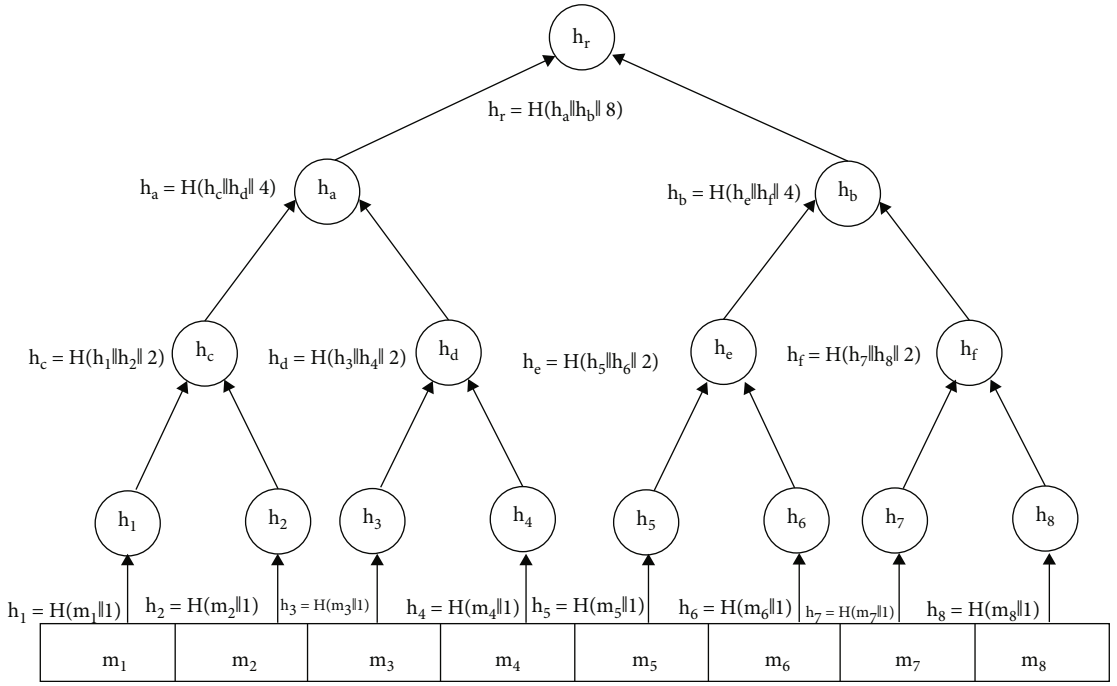

Figure 1: An example of RMHT.

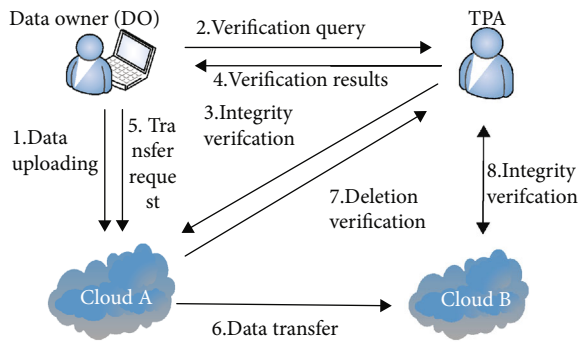

Figure 2: A provable data migration model.

A should be checked first. If the data is complete, DO send a data transfer request to cloud A, and then cloud A transmits the data to cloud B. When cloud B receives the data, it checks whether the data is integrated or not. Otherwise, DO requires cloud A to resend the data. DO uses integrity checking algorithms to ensure that the whole file transferred to cloud B is correct

(4) DeletCheck: run by DO and clouds A and B. When data is successfully transferred to cloud B, Cloud A discards the migrated data. In order to ensure that cloud A executes the data deletion operation correctly and successfully deletes data, DO sends a query request to TPA, and TPA sends a challenge to cloud A to detect whether cloud A has successfully deleted data. After cloud A accepts the challenge, it computes a data deletion proof and sends it to TPA, which verifies the proof and returns the result to DO. Based on results, DO knows whether the transmitted data was successfully discarded

(5) IntegCheck

It is a data integrity check algorithm, run by DO, TPA, and cloud, to check whether the data on the cloud is integrated. DO generates a validation query and sends it to the TPA, and TPA launch a challenge to cloud to check the integrity of the outsourced data. When accepting the challenge, the cloud computes the proof using the stored data and returns it to TPA, which checks validity of the proof and sends the test result back to DO.

In this framework, DO uses TPA to perform data ownership and data deletion verification, which implement public verification. DO also makes a verification without the help of TPA. Each data check is independent of TPA and DO.

The detailed steps of each stage are in reference [9]. Therefore, no detailed descriptions are given.

5. Hierarchical Data Authentication Mode

5.1. Basic Principle. When authenticating a large number of continuous files on the cloud, the common methods are to authenticate one by one or at random. Unfortunately, the former authentication overhead is too much, while the latter has high rate of missed detection. To solve this problem, a hierarchical verification mode (HVM) is proposed in this paper.

The basic idea of HVM is to select a number of files or data blocks for coarse-grained challenge response authentication and then to carry out fine-grained authentication in the adjacent area. The working process is as followed: firstly, the first layer authenticated data blocks are determined

according to the initial access granularity, and then the coarse-grained authentication at the first layer is carried out.

If the authentication is successful, the data blocks in the adjacent area are considered as being integrated, without the need of authentication. If the authentication fails, the adjacent data blocks will be authenticated at the second level with finer granularity. As the verification fails again, a third level of authentication is performed on adjacent blocks at finer granularity, and so on, until the minimum detection granularity is reached.

5.2. Detail Steps. For multiple files continuously stored on the cloud, the steps of HVM are as followed:

(1) Set a detection flag bit A for each file on clouds, with an initial value of 0

(2) The number of files to be authenticated on the cloud server is determined by TPA, and the initial detection granularity (i.e., the number of file intervals) X is determined

(3) Divide all the files to be detected by X equally, and equal diversion point is the initial detection point

(4) The agent challenges each file at the first level checkpoint on the server

(5) The cloud server judge the flag bit of the file. If the flag bit is 1, the server will skip this detection. Otherwise, the cloud server sets the flag bit to be 1 and generates a response proof to this challenge, and sends the proof to agent

(6) The agent verifies the proof sent by TPA. If the authentication is successful, go to step (10)

(7) If the authentication fails, judge whether X is the minimum detection granularity. If so, execute step (10); otherwise, $X = X/2$

(8) The agent challenges files separated by X before and after the file on cloud server

(9) Execute step (5)

(10) Verification is over

6. Hierarchical Provable Data Migration Method

In order to solve the problem of high verification overhead when massive files are migrated between clouds, HVM is applied to the integrity verification of cloud data migration. Combined HVM with the framework of integrity authentication of cloud data migration [9], a hierarchical provable data migration method is proposed, which includes two algorithms: H-Transfer (hierarchical provable data migration method) and H-IntegCheck (hierarchical provable integrity detection method).

The application scenario of this method is the data migration model in Figure 2. Data storage, verification, and migration are carried out on clouds A and B. The operation objects are the batch files continuously stored on clouds A and B. To facilitate verification, DO sets the detection flag bit for each file (the initial value is 0, indicating that the file has not been authenticated) and then determines the initial detection granularity X and the minimum detection granularity $X_{\min}$.

6.1. H-IntegCheck Algorithm Description. The scenario of this algorithm is to verify the data integrity of massive files continuously stored on cloud A or cloud B, which is described as followed:

Let N be the number of files to be authenticated on clouds. Determine the first layer detection files with N and X and then execute the following operations for each file.

(1) To verify the data integrity on cloud A or cloud B, TPA chooses a subset S and a random number $\theta \in Z_q^*$ for the current file from $[1, n]$, where n is the number of blocks in the current file. Then, TPA sends the challenge message $\text{chal} = \{S, \theta\}$ to cloud server

(2) After receiving chal from TPA, the cloud first generates $\{p_i = \theta^i \bmod q\}, i \in S$. Then, the cloud server generates $y = f_{\vec{A}}(\theta)$, where $\vec{A} = \{0, 0, \sum_{i\in S} p_i m_{i,0}, \cdots, \sum_{i\in S} p_i m_{i,s-1}\}$. The cloud server divides the polynomial $f_{\vec{A}}(x) - f_{\vec{A}}(\theta)$ by $x - \theta$, and the coefficients vector of the resulting polynomial is denoted by $\vec{\omega}$. $\vec{\omega} = (\omega_0, \omega_1, \cdots, \omega_{s+1})$. The cloud server computes $\varphi = \prod_{j=2}^{s+1} (g^{a^j})^{\omega_j}$. Finally, the cloud obtains $\sigma = \prod_{i\in S} \sigma_i^{p_i}$ and sends the proof. $P = \{\sigma, \varphi, y, sig_{ssk}(H(R)), \{H(m_i \| 1), \Omega_i\}_{i\in s}\}$ to TPA, where $\{\Omega_i\}_{i\in s}$ is the auxiliary authentication information of block i

(3) When TPA receives P from the cloud, it will judge whether the flag bit of the corresponding file is 0. If not, the authentication of the current file ends. Otherwise, TPA first generates R with $\{H(m_i \| 1), \Omega_i\}_{i\in s}$ and then authenticates the validity of R with $sig_{ssk}(H(R))\}$, and the corresponding mark position is 1. If the verification fails, TPA aborts and return false, then go to (4). Otherwise, TPA computes $= u^{\sum_{i\in S}\{p_i H(\text{name})\}}$, and then TPA checks whether the following equation holds

$$e(\eta, \delta) \cdot e\left(\varphi, \gamma \cdot \delta^{-\theta}\right) \stackrel{?}{=} e(\sigma, g) \cdot e(\delta^{-y}, g). \quad (1)$$

If yes, it means that the data on cloud A or cloud B are integrated; otherwise, output *false*.

When $X/2$ is no less than $X_{\min}$, the $X/2^{\text{th}}$ file before this file and the $X/2^{th}$ file after this file are taken as the file in next hierachy verification process, go to (1).

The process of integrity verification for H-IntegCheck is illustrated in Figure 3.

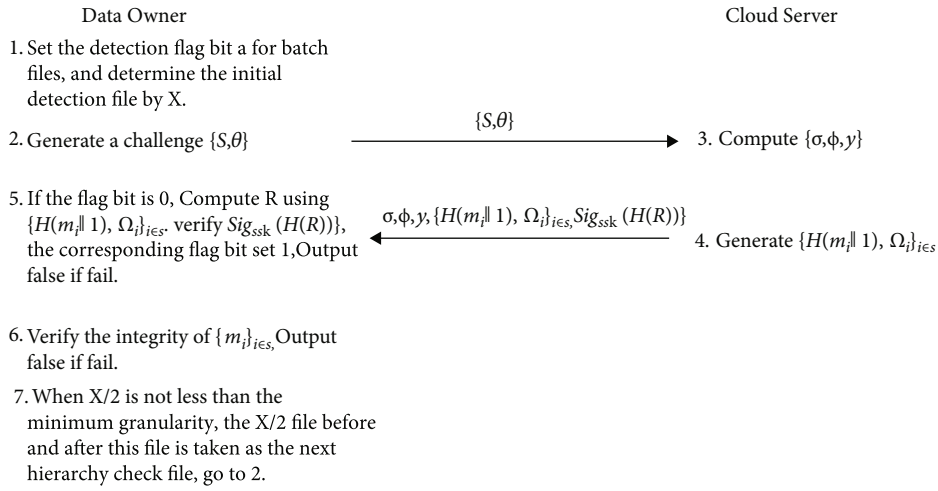

Figure 3: The working process of H-IntegCheck.

6.2. H-Transfer Algorithm Description. The scenario of this algorithm is to transfer massive files of continuous stored between cloud A and cloud B and to implement the secure transmission of provable data integrity. Details are illustrated as followed.

Determine the first layer detection files with N and X and then execute the following operations for each file.

(1) When DO transfers part or all of the current file F from cloud A to cloud B, it first calls H-IntegCheck algorithm to check the data integrity on cloud A. If the file is corrupted, the user aborts check and returns *false*, then go to (5). Otherwise, DO retrieves τ from cloud A and verifies the signature using *spk*. If the signature is invalid, DO aborts check and output *false*, and go to (5). Otherwise, DO obtains the positions of sentinels *PF* by decrypts C with k

(2) DO generates the block index Ψ to be transferred and computes the data migration request $Q = fname\|\Psi\|sign_{ssk}(fname\|\Psi)$ and a tag, $\tau^* = fname\|n^*\|C^*\|Sign_{ssk}(fname\|n^*\|C^*)$, where n^* is the number of the data blocks to be transferred, and C^* is the encryption of positions of the sentinels in the block to be transferred. And then DO sends (Q, τ^*) to cloud A

(3) Cloud A verifies the signature $Sign_{ssk}(fname\|\Psi)$ and returns *false* if the signature is invalid. Otherwise, cloud A sends (Q, τ^*, ϵ) to cloud B, with the data blocks $\{m_i\}_{i\in\Psi}$ the corresponding authentication tags $\{\sigma_i\}_{i\in\Psi}$ and $AAI\Omega^*$ of RMHT. $AAI\Omega^*$ is the sibling nodes on the path from the leaves to the root of MHT

(4) When receiving the messages from cloud A, cloud B verifies the signatures $Sign_{ssk}(fname\|\Psi)$ and $Sign_{ssk}(fname\|n^*\|C^*)$. If either is invalid, cloud B aborts check and returns *false*. Otherwise, cloud B go on to check each block tag σ_i, for $i \in \Psi$ as $e(u^{n*H(fname)}\prod_{j=0}^{s-1}(g^{\alpha^{j+2}})^{\sum_{i\in\Psi}m_{ij}}, \delta) = ^{?} e(\prod_{i\in\Psi}\sigma_i, g)$. If the equation does not hold, cloud B aborts check and returns *false*; otherwise, it utilizes the AAI Ω^*, and data blocks m_i, $i \in \Psi$, to reconstruct RMHT to obtain root R^*

(5) Finally, cloud B verifies whether ϵ is the signature of R^* by using spk. If the verification fails, cloud B aborts check and returns *false*; otherwise, it stores $(\tau^*, \{m_i\}_{i\in\Psi}\Omega^*, \epsilon)$ on its server and returns $(fname, \Omega^*, \epsilon)$ to DO to affirm the data transfer is successful. The data owner uses Ω^* to reconstruct RMHT to compute its root $\widehat{R}$ and checks if it is valid by verifying $\epsilon = ^{?} Sign_{ssk}(\widehat{R})$. TPA obtains root node h^* by $\{i, \Omega_i\}_{i\in\Psi}$ to reconstruct RMHT, and then TPA verifies $h^* = ^{?} h_R^*$ to confirm if the data is deleted. If authentication fails, TPA outputs false and returns to DO

(6) When $X/2$ is no less than X_{min}, the $X/2^{th}$ file before this file and the $X/2^{th}$ file after this file are taken as the file in next hierachy verification process and then go to (1). The integrity verification process of H-Transfer is shown in Figure 4

7. Security Analysis

7.1. Provable Data Possession and Provable Data Transfer Analysis. The proposed hierarchical provable data method includes H-IntegCheck and H-Transfer algorithms, which optimized algorithms IntegCheck and Transfer. The security design goals of H-IntegCheck are provable data possession. It ensures that a polynomial time adversary *Adve* cannot successfully pass the verification with a forged proof, unless it can guess all the missing blocks. By using $\{H(m_i), \Omega_i\}_{i\in S}$ and $sig_{ssk}H(R)$ returned from cloud server, the user can authenticate the validity of R and auxiliary authentication information of m_i. If $\{\Omega_i\}_{i\in S}$ has changed, it cannot pass the signature verification. If the signature scheme is existentially unforgeable, then the $\{\Omega_i\}_{i\in S}$ and $sig_{ssk}H(R)$ are preserved completely. For the integrity of the blocks, the

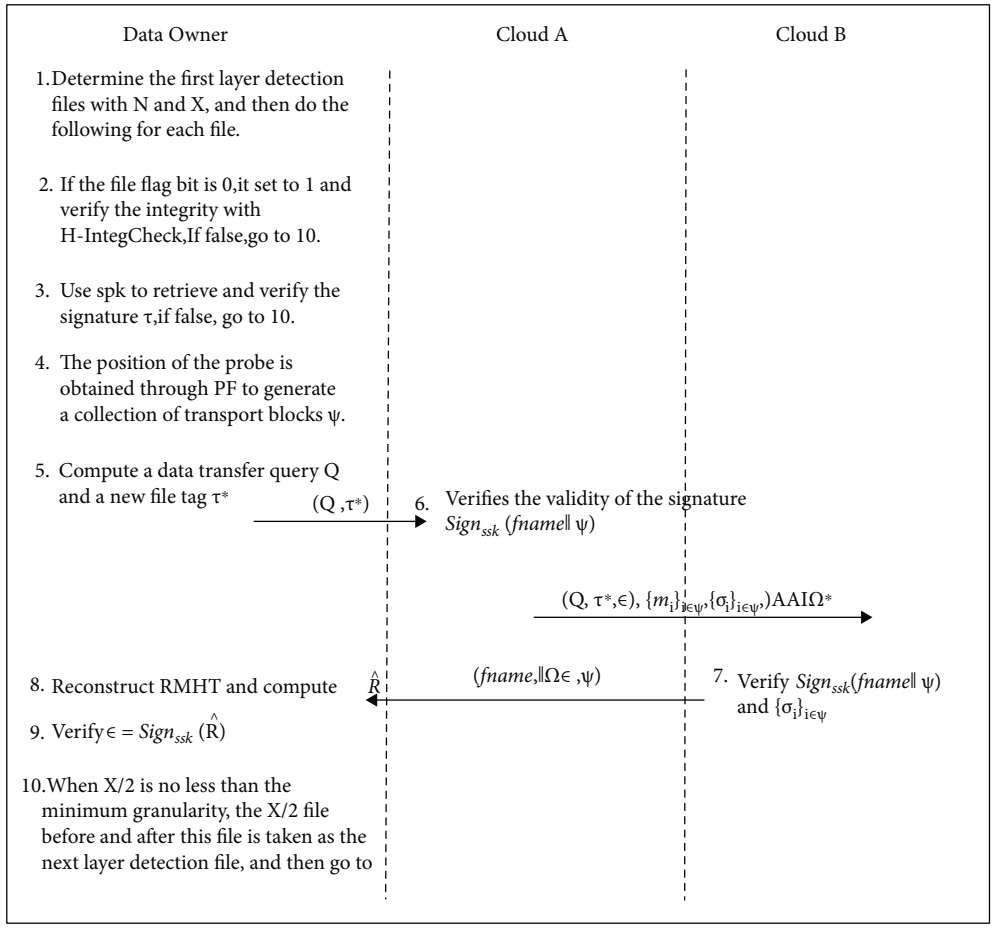

Figure 4: The working process of H-Transfer.

adversary cannot obtain a forged proof which can pass the verification in polynomial time.

The security design goals of H-Transfer is provable data transfer. It ensure the data can be transferred successfully. The user firstly checks the data integrity on cloud A before the transfer. If the file is corrupted, the user will investigate legal liability of cloud A. By verifying the signature in τ, user is convinced the ciphertext of table PF is intact. In order to ensure the data integrity during the data transfer, cloud B checks the integrity of each block by aggregatable verification of tags. If the integrity check is successful, then the data are transferred successfully to the cloud B. If only a part of data are received by cloud B, it will reject it and ask cloud A to retransmit the data. For the transfer request contains the user's signature, the adversary cannot forge it in polynomial time; hence, the data transfer operation is executed under the delegation of the user. After the data has been transferred completely, cloud B will send message to acknowledge the user the success of the data transfer. By utilizing the information returned from cloud B, the user checks the correctness of the root. If the verification is successful, the data transfer is successful.

7.2. Missing Report Rate Analysis. For H-IntegCheck and H-Transfer algorithms both select part of the files for verification, some corrupted files may not be detected, which may result in missing detection. The following is a brief analysis of the miss detection rate by taking H-transfer algorithm as an example and Transfer algorithm as the comparison object.

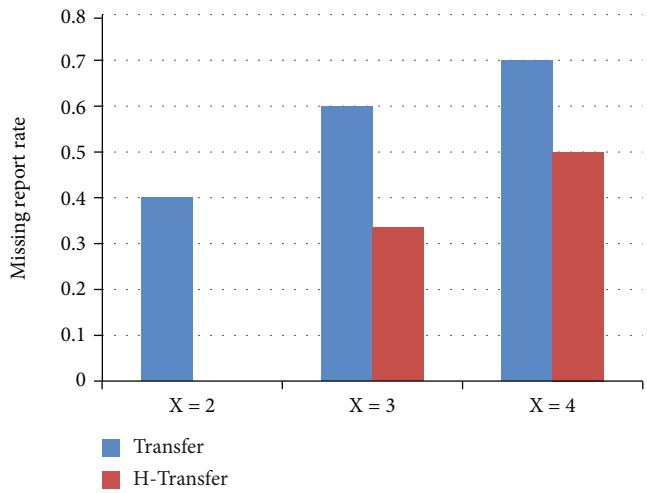

Figure 5: Missing report rate with correlation degree of 2.

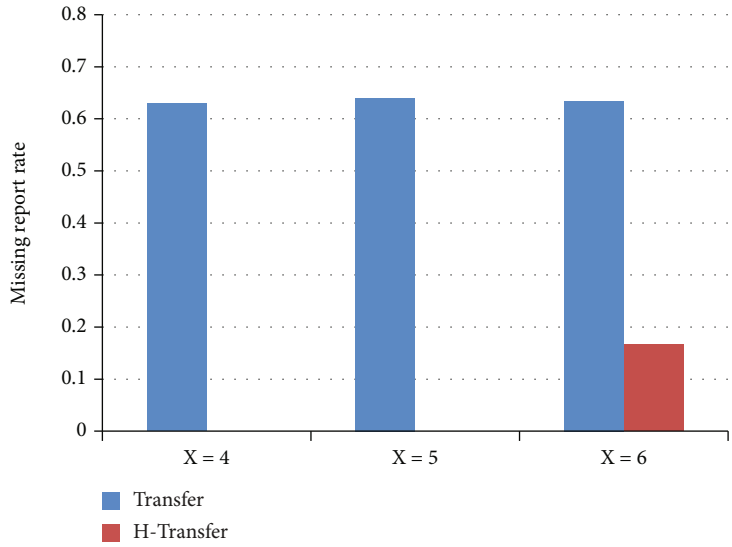

Figure 6: Missing report rate with correlation degree of 5.

To complete the analysis, Transfer and H-transfer algorithms are implemented based on PBC Library on Ubuntu14. There are 500 files to be verified, of which 10% files are corrupted. The correlation degrees are 2 and 5, respectively. The correlation degree means the number of files corrupted continuously:

7.2.1. Analysis of Missing Report Rate with a Correlation Degree of 2. The correlation degree is 2, and the missing report rate with the initial detection granularities of 2, 3, and 4 are shown in Figure 5. Compared with Transfer, the missing report rate of H-transfer is significantly decreased. When the initial detection granularity is 2, H-Transfer verifies all files at one interval, and all the error files with correlation degree 2 can be detected. Therefore, the missing report rate is 0. Transfer has a certain rate of missed detection (40%). When the initial detection granularity is 3 or 4, for the detection interval is greater than or equal to 2, there may be two consecutive false files missing detection, and so the missing report rate increase to 33.6% and 50%, respectively. However, the missing detection rate of H-Transfer is 33.6% and 50%, which is reduced by 44% and 28.6%, respectively, compared with Transfer.

7.2.2. Analysis of Missing Report Rate with a Correlation Degree of 5. The correlation degree is 5, and the missing report rate with the initial detection granularity of 4, 5, and 6 is shown in Figure 6. Compared with Transfer, the missing report rate of H-transfer is also significantly decreased. When the initial detection granularity is 4 or 5, H-Transfer verifies all files at intervals of 3 or 4. All the wrong files with a correlation of 5 can be detected; so, the missing report rate is 0. When the initial detection granularity is 6, 5 consecutive error files may be undetected. The missing report rate increases to 16.6%. However, compared with Transfer, its missing report rate is decreased by 73%. According to the above analysis, for H-transfer, when the initial detection granularity is less than or equal to the correlation degree, missing report rate is the lowest, but more detection times are needed. When the initial detection granularity is greater than the correlation degree, there are fewer detection times, but higher missing report rate. In the case of same verification overhead, H-Transfer has a lower missing report rate than Transfer.

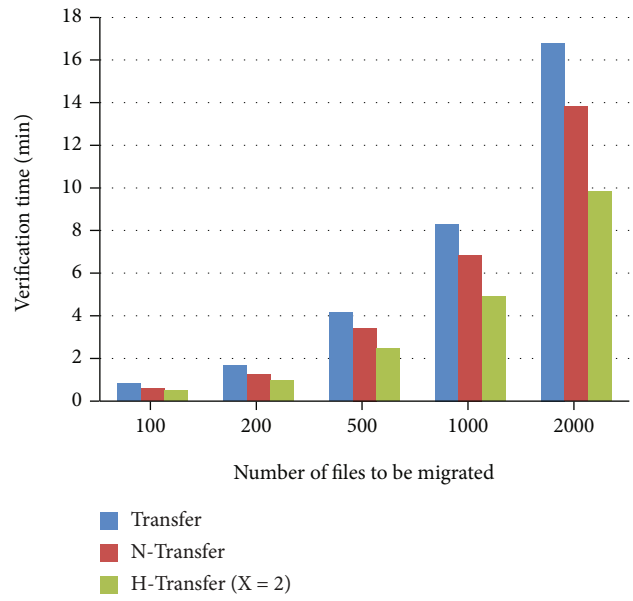

Figure 7: Verification time with different numbers of files to be migrated.

8. Performance Evaluation

The performance of the proposed method is evaluated below. The proposed method contains H-Transfer and H-IntegCheck processes. Because H-Transfer and H-IntegCheck work in a similar way, H-transfer is taken as an example to evaluate their performance. The comparison

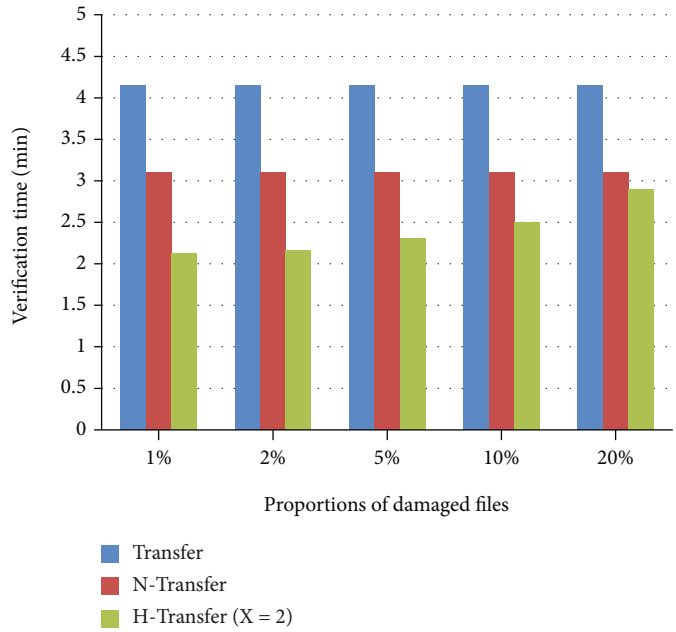

Figure 8: Verification time with different proportions of corrupted files.

object is Xue's [9] transfer (Transfer) process and Liu's [14] new transfer (N-Transfer) process with modular multiplication operation and modular exponentiation operation. The experiment is run on a PC with Intel i5 CPU and 4G memory. Transfer, N-Transfer, and H-transfer processes are implemented based on PBC library. The scenario is that cloud A migrates massive files to cloud B as shown in Figure 2, and the files are corrupted in a certain proportion. Transfer, N-Transfer, and H-transfer processes are used to verify the migration files, respectively, and the verification time is selected as the performance evaluation index:

8.1. Performance Evaluation with Different Numbers of Files to be Migrated. In this evaluation, cloud A transfers 100, 200, 500, 1000, and 2000 continuously files to cloud B, respectively. The proportions of corrupted files are 10%. For H-Transfer, both initial detection granularity X and correlation degree are 2. The performance evaluation results are shown in Figure 7. As shown in the figure, the verification time of N-Transfer decreased 21.3% in average than Transfer. For N-Transfer only uses modular multiplication and modular exponential operations, and the complex operations (such as bilinear mapping) do not use, so it is more efficient than Transfer. Moreover, the verification time of H-Transfer decreased 41.5% and 25.3% in average than Transfer and N-Transfer, respectively. For H-Transfer is to efficiently choose high probability of corrupted files for authentication, the overhead is lower than N-Transfer only with modular multiplication and modular exponential operations.

8.2. Performance Evaluation with Different Proportions of Corrupted Files. In this evaluation, the proportions of corrupted files are 1%, 5%, 10%, 15%, and 20%, respectively. Cloud A transfer 500 continuously files to cloud B. Similar with evaluation (1), both initial detection granularity and correlation degree are 2. The performance evaluation results are shown in Figure 8. For Transfer and N-Transfer, all migrated files need to be authenticated, and the verify overhead is the authentication overhead of all migrated files. Therefore, changes in proportion of corrupted files have little impact on the authentication performance. For H-Transfer, the verification time is lower than Transfer and N-Transfer obviously. Moreover, the performance improvement slowly decreases as the proportion of corrupted files increases. Compare with Transfer and N-Transfer, the verification time of H-Transfer decreases by 48.7% and 31.3%, respectively, when the proportion is 1%, and the verification time decreases by 30% and 6.5%, respectively, when the proportion increases to 20%. The average verification time decrease by 42.2% and 22.6%, respectively. The reason is that with the increase of the proportion, the number of files verified by H-Transfer also gradually increases.

8.3. Performance Evaluation with Different Initial Detection Granularity. In this evaluation, the initial detection granularity X are 2, 3, 4, 5, and 6, respectively. Similar to evaluation (2), cloud A transfer 500 continuously files to cloud B, and the correlation degree is 2. The performance evaluation results are shown in Figure 9. As shown in the figure, the change of initial detection granularity has little impact on the authentication performance for Transfer and N-Transfer. For H-Transfer, the verification time is lower than Transfer and N-Transfer obviously. Moreover, the verification time decreases slowly as initial detection granularity increases. Compare with Transfer and N-Transfer, the verification time of H-Transfer decreases by 39.8% and 19.3% as X is 2 and decreases by 46.5% and 28.4% as X is 6. The average verification time is decrease by 43.9% and 24.9%. The reason is that with the initial detection granularity increases,

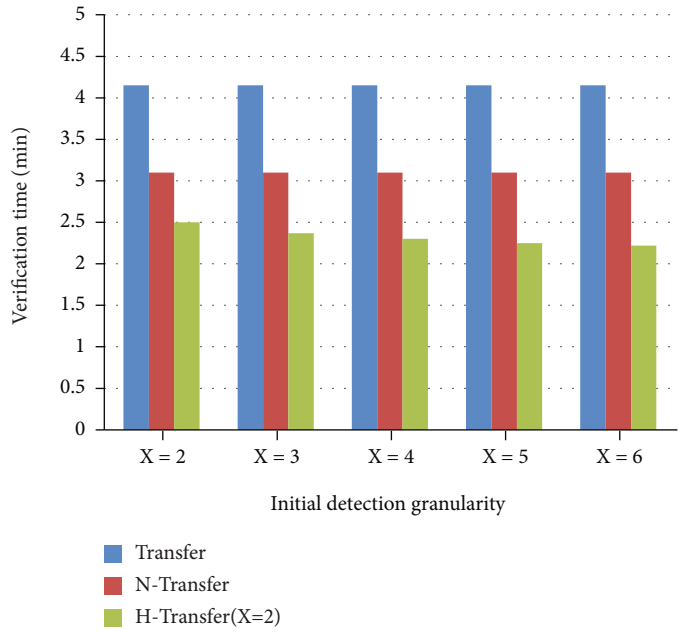

Figure 9: Verification time with different initial detection granularity.

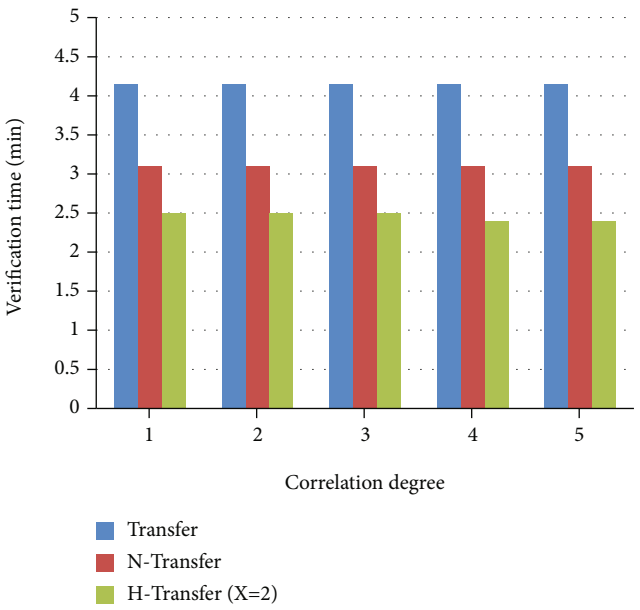

Figure 10: Verification time with different correlation degree.

the number of files verified of H-Transfer gradually decreases.

8.4. Performance Evaluation with Different Correlation Degrees. In this evaluation, the correlation degree is 1, 2, 3, and 5, respectively. Similar to evaluation (3), cloud A transfer 500 continuously files to cloud B, and the initial detection granularity is 2. The performance evaluation results are shown in Figure 10. As shown in the figure, the change of correlation degree has little impact on the authentication performance for Transfer and N-Transfer. For H-Transfer, the verification time is lower than Transfer and N-Transfer obviously. Moreover, the verification time decreases very slowly as correlation degree increases. Compare with Transfer and N-Transfer, the verification time of H-Transfer decreases by 39.8% and 19.4% as correlation degree is 1 and decreases by 42.2% and 22.6% as correlation degree is 5. The average verification time is decrease by 40.7% and 20.6%. The reason is that with the correlation degree increases, the number of files verified of H-Transfer also slowly decreases.

9. Conclusions

This paper proposed a hierarchical verification mode and a hierarchical provable data migration method. The former is an optimized selection authentication mode, which can be combined with multiple data integrity authentication methods to improve the detection efficiency. The latter improves the transfer and integcheck algorithms of the reference [9]. The analysis and evaluations proved that the proposed method can effectively decreased the integrity authentication time of massive files migration between clouds. Therefore, this method can provide better authentication performance.

In the future work, the other algorithms in framework of integrity authentication of cloud data migration will be further optimized, and we will investigate migration security and performance of non-continuous massive files to reduce verification overhead for improving the performance of file migration while guaranteeing security.

Data Availability

Experimental data is randomly generated on simulation tools.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by Guangxi Natural Science Fund (No. 2018GXNSF AA138209), Science Research Start Fund of Shanghai University of Technology (No. 39120K196002-A06), and Science Research Start Fund of Guilin University of Technology (No. GUTQDJJ2017).

References

[1] C. Wang, D. Wang, G. Xu, and D. He, *Efficient Privacy-Preserving User Authentication Scheme with Forward Secrecy for Industry 4.0*, SCIENCE CHINA: Information Sciences, 2020.

[2] C. Wang, D. Wang, T. Yi, X. Guoai, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, 2020.

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

[4] S. K. Nayak and S. Tripathy, "SEPDP: secure and efficient privacy preserving provable data possession in cloud storage," *IEEE Transactions on Services Computing*, vol. 1, 2018.

[5] T. Jiang, X. Chen, J. Li, D. S. Wong, J. Ma, and J. K. Liu, "Towards secure and reliable cloud storage against data re-outsourcing," *Future Generation Computer Systems*, vol. 52, no. 11, pp. 86–94, 2015.

[6] Y. J. Ren, J. Shen, J. Wang, J. Han, and S. Y. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, vol. 16, no. 2, pp. 317–323, 2015.

[7] A. Juels, S. Burton, and K. Jr, "PORs: Proofs of Retrievability for Large Files," *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS, 07)*, 2007, pp. 584–597, Alexandria, Virginia, USA, October 2007.

[8] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," in *Proceeding (S) of ACM Workshop on Cloud Computing Security*, pp. 43–53, Chicago, USA, 2009.

[9] 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, pp. 46–54, 2017.

[10] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *4th International Conference on Security and Privacy in Communication Networks*, pp. 1–10, Istanbul, Turkey, September 2008.

[11] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *In 16th ACM CCS*, pp. 213–222, Chicago, Illinois, USA, 2009.

[12] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," in *The 28th International Conference on Distributed Computing Systems, 2008. ICDCS '08*, Beijing, China, June 2008.

[13] M. Etemad and A. Küpçü, "Transparent, Distributed, and replicated dynamic provable data possession," in *Applied Cryptography and Network Security*, vol. 7954, pp. 1–18, Springer, Berlin Heidelberg, April, 2013.

[14] 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. 5, 2019.

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

[16] H. Wang, D. He, A. Fu, Q. Li, and Q. Wang, "Provable data possession with outsourced data transfer," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[17] W. I. Khedr, H. M. Khater, and E. R. Mohamed, "Cryptographic accumulator-based scheme for critical data integrity verification in cloud storage," *IEEE Access*, vol. 7, no. 6, pp. 65635–65651, 2019.

[18] R. Almarwani, N. Zhang, and J. Garside, "An effective, secure and efficient tagging method for integrity protection of outsourced data in a public cloud storage," *PLoS One*, vol. 15, 2020.

[19] H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'08)*, 2008pp. 90–107, Berlin, Melbourne, Australia, 2008.

[20] C. Wang, Q. Wang, K. Ren, and W. Lou, *Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing29th IEEE INFOCOM*, pp. 525–533, San Diego, CA, USA, March 2010.

[21] Y. Yong, H. A. Man, and G. Ateniese, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 4, pp. 767–778, 2017.

[22] H. Huiying, Y. Jia, and Z. Hanlin, "Enabling secure auditing and deduplicating data without owner-relationship exposure in cloud storage," *Cluster Computing*, vol. 21, pp. 1849–1863, 2018.

[23] X. Zhang, J. Zhao, C. Xu, H. Li, H. Wang, and Y. Zhang, "CIPPPA:Conditional identity privacy preserving public auditing for cloud-based WBANs against malicious auditors," *IEEE transactions on cloud Computing*, vol. 9, 2019.

[24] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS:Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[25] S. Jegadeesan, M. Azees, N. Ramesh Babu, U. Subramaniam, and J. D. Almakhles, "EPAW: efficient privacy preserving anonymous mutual authentication scheme for wireless body area networks (WBANs)," *IEEE Access*, vol. 8, no. 3, pp. 48576–48586, 2020.

[26] A. P. Mohan, A. R. Mohamed, and A. Gladston, "Merkle tree and Blockchain-based cloud data auditing," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 10, 2020.

[27] Y. Yu, J. Ni, M. H. Au, H. Liu, H. Wang, and C. Xu, "Improved security of a dynamic remote data possession checking protocol for cloud storage," *Expert Systems with Applications*, vol. 41, no. 17, pp. 7789–7796, 2014.

[28] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud

Computing," *Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS'09)*, , pp. 355–370, Springer-Verlag, Berlin, 2009.

[29] S. Qiu, D. Wang, X. Guoai, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for Mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, 2020.

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

[31] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1703–1713, 2014.