JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SECURE DATA TRANSFER AND DELETION FROM CLOUD COMPUTING'S BY USING COUNTING BLOOM FILTER

[1]V.Pavankumar, [2]Dr C Prakasa Rao
[1]M.Tech student, [2]Professor
[1]Computer Science Engineering,
[1]Visvodaya Engineering College, Kavali, India

Abstract : Because of the rapid development of cloud storage, an increasing number of data owners opt to outsource their data to a cloud server, which can significantly lower the local storage overhead. Because different cloud service providers provide varying levels of data storage quality, such as security, dependability, access speed, and pricing, cloud data transfer has become a key requirement for data owners when switching cloud service providers. As a result, how to securely migrate data from one cloud to another and completely remove transferred data from the original cloud becomes a top priority for data owners. In this research, a new counting Bloom filter-based technique is developed to overcome this problem. The suggested plan can realise permanent data deletion in addition to secure data transport. Furthermore, the suggested approach can satisfy public verifiability requirements without the need for a reliable third party. Finally, create a simulation implementation to show how effective and useful the recommendation is.

***IndexTerms* - Cloud storage, Data deletion, Data transfer, Counting Bloom filter, Public verifiability.**

I. INTRODUCTION

Large-scale dispersed storage resources, computer resources, and network bandwidths are all connected through cloud computing, a new and very promising paradigm for computing[1,2]. These assets allow it to offer a wide range of top-notch cloud services to tenants.

The services (particularly cloud storage services) have been widely used because of its alluring advantages[3,4], allowing resource-constrained data owners to outsource their data to a cloud server, significantly reducing the local storage overhead of the data owners[5,6]. The estimate by Cisco[7] predicts that by 2022, there will be 3.6 billion Internet users, or nearly 55 percent of the world's of them will make use of cloud storage.

A growing number of businesses (such as Microsoft, Amazon, and Alibaba) are offering cloud storage services to data owners with varying costs, levels of security, and access times due to the market's bright future. The data owners may switch cloud storage service providers to benefit from a more suited cloud storage service. Deleting the transferred data from the original cloud after migrating their outsourced data from one cloud to another. By the end of 2021, cloud traffic is anticipated to make up 95% of all traffic, with traffic between separate cloud data centres accounting for roughly 14% of the total cloud traffic, according to Cisco[7]. From the perspective of the data owners, it is foreseeable that the outsourcing of data transfer will become a key requirement..

An external data transfer software called Cloudsfer[8] has been developed to achieve secure data migration. It uses cryptographic algorithms to prevent data privacy disclosure during the transfer process. However, there are still a few security issues with how the migration and deletion of cloud data is handled. To start, the cloud server may just move a portion of the data in order to save network capacity, or it may even deliver some unrelated data in order to defraud the data owner[9]. Second, certain data blocks may be lost during the transmission process due to network instability.

The sent data blocks could be destroyed in the interim by the adversary[10]. As a result, throughout the migration process, the transferred data may become contaminated. Not to mention, the original cloud server may maliciously retain the transferred data for uncovering the benefits that are implied[11]. From the perspective of the data owners, the reserve of the data is unanticipated. In conclusion, the cloud storage service is cost-effective but necessarily faces significant security issues, particularly with regard to safe data transfer, integrity verification, and verifiable destruction. If these issues are not properly resolved, the public may be prevented from accepting and using cloud storage services.

CONTRIBUTIONS

In this work, we examine the issues with safe data transfer and deletion in cloud storage with a particular emphasis on achieving public verifiability. Then, we suggest a counting Bloom filter-based system that not only enables data deletion that can be independently verified by the public, but also enables provable data movement between two distinct clouds. The data owner and the target cloud server can verify the returned transfer and deletion evidences to identify malicious acts if the original cloud server does not migrate or erase the data in an honest manner. In contrast to the alternatives, our suggested approach does not call for the use of a trusted third party (TTP). Additionally, we demonstrate through security analysis that our new approach may fulfil the intended design objectives. Finally, the simulation experiments demonstrate the viability and effectiveness of this innovative suggestion.

comparable works Verifiable data erasure has been thoroughly researched for a very long time, leading to numerous solutions[12–18]. "Xue et al."[19] investigated the objective of secure data deletion and proposed a key-policy attribute-based encryption technique that may accomplish data finegrained access control and ensured deletion. They achieve data deletion by eliminating the attribute, and they achieve verifiability using the Merkle hash tree (MHT), however their plan requires a reliable source of information. Associated deletion strategy for multi-copy (ADM), created by Du et al. [20], employs pre-deleting sequence and MHT to achieve data integrity verification and proven deletion.However, to control the data keys, their system also needs a TTP.

A Blockchain-based cloud data deletion technique was introduced in 2018 by Yang et al. [21], in which the cloud performs deletion operations and posts the corresponding deletion proof on Blockchain. The deletion proof can then be verified by any verifier in order to check the deletion result. Additionally, they remove the hindrance of needing a TTP. All of these strategies can verify data erasure, but they can't transfer data securely.

Numerous approaches have been suggested[22–26] for moving data between clouds and deleting the transferred data from the original cloud. A Provable data possession (PDP) approach that can also facilitate secure data migration was introduced by Yu et al. in 2015 [22]. To the best of our knowledge, their plan is the first to effectively handle data transfer between two clouds, but it is ineffective for data deletion since they achieve deletion by re-encrypting the transferred data, which necessitates a lot of information from the data owner.A verifiable data movement strategy was created by Xue et al.

which is distinguished by verifiable deletion and PDP. The data owner can use the PDP protocol to validate the data's integrity and the Rank-based Merkle hash tree (RMHT) to confirm the deletion's outcome. However, Liu et al. [25] pointed out that the system in Ref. [24] has a security weakness, and they created an improved method that can address the flaw. Yang et al. [26] adopted vector commitment in 2018 to create a new data transfer and deletion scheme that allows the data owner to independently verify the results of the transfer and deletion without the use of TTP. Additionally, their plan may carry out data integrity verification on the intended cloud.

Organizations: The remainder of this essay is structured as follows: The preliminaries are described in Section II, and the problem statement is described in Section III.In Section IV, we introduce our innovative scheme. Then, in Section V, the developers analyse security, and in Section VI, offer an evaluation of performance using simulated trials. In Section VII, the paper's conclusion and its future work are presented.

II. PRELIMINARIES

A set can be tested to see if it contains a certain element using the space-saving data structure known as the bloom filter (BF)[27]. The cost of inserting BF is continual time overhead. regardless of how many elements the set and BF contain, an element or determine whether an element is a part of the set.

A BF can be thought of as a bit array of length m with k hash functions $h_i(.)=\{0,1\}^* \rightarrow \{0,1,\ldots\ldots,m\}$. We must set the group of k bits to 1 in order to insert an element; the placements of these bits are defined by the hash values $h_1(x)\ldots\ldots\ldots,h_k(x)$. The same hash calculations are used to construct membership tests, and they are successful if all of the corresponding places are one, as illustrated in Fig. 1. Be aware that there is a false positive in the BF, which indicates that even if all k of the bits associated to *w* are ones, *w* has a low likelihood of not being a member of the set. To lessen the probability, we can set the right parameters, such as the number of hash functions *k*, the length of the BF m, and the number of components n [28]. If the parameters are appropriate, the chance will also be so small as to be negligible [29].

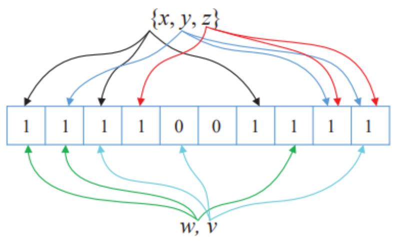

Fig. 1. Example of Bloom filter

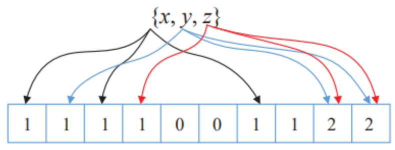

Fig. 2. Example of counting Bloom filter

Additionally, BF is unable to remove a component from the data set. The Counting Bloom filter (CBF) is introduced to address this problem [30,31]. As a BF version, CBF substitutes each "bit" position with a counter cell count, as seen in Fig. 2. The *k* related counters must be increased by one in order to insert element y; the counters' indexes are also defined by the hash values $h_1(y)$, $h_2(y)$, , and $h_k(y)$. The element deletion action, however, merely reduces the k related counters by one.

III. PROBLEM STATEMENT

The system framework, security issues, and security objectives are briefly presented in the sections that follow.

1. System framework

In the end, one can want to achieve reliable data erasure from cloud storage and verifiable data transfer between two different clouds. Thus, as illustrated in Fig. 3, three entities are incorporated in our new construction. In our example, the owner of the large-scale data who is under resource constraints could outsource his data to cloud server A in order to significantly minimise the local storage overhead. Besides,

The owner of the data may request that certain data be deleted from the storage medium or moved from cloud A to cloud B. Cloud storage is made available to the data owner by clouds A and B. We suppose that cloud A is the initial cloud, from which certain data must be transported to cloud B, where it will be deleted. However, for financial considerations, cloud A might not carry out these processes honestly. Furthermore, since cloud A and cloud B are owned by distinct companies, we think that they won't work together to deceive the data owner. Consequently, the two clouds will each adhere to the procedure on their own.

Additionally, we presumptively assume that the target cloud B won't maliciously disparage the original cloud A.

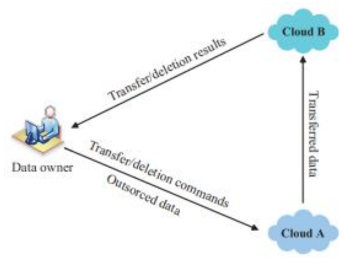

Fig. 3. The system framework

2. Design goals

The strategy should achieve the three objectives listed below.

1) Information security. The file that was outsourced can have some sensitive information in it that needs to be kept hidden. Therefore, the data owner must employ secure algorithms to encrypt the file before uploading it to the cloud server in order to safeguard the confidentiality of the data.

2) Data reliability. It's possible that the cloud A will just transfer a portion of the data or will send the cloud B some unrelated data. In addition, the data transmission procedure itself could contaminate the data. Therefore, to ensure that the transferred data is complete, the data owner and cloud B should be able to check the integrity of the data.

3) Publicly verifiable. The data may not be faithfully deleted or transferred from cloud A to cloud B. Therefore, from the perspective of the data owner, the verifiability of the transfer and deletion results should be satisfied.

IV. THE PROPOSED SCHEME

We thoroughly describe our new system in this section. Keep in mind that the owner of the data must be verified by the cloud storage service provider. For the sake of simplicity, we'll suppose that the owner of the data has successfully completed the identifying process and obtained legal occupancy of clouds A and B.

1. Overview

Similar to Ref.[26], our scenario aims to enable verifiable data transfer and destruction. The primary procedures are depicted in Fig. 4. The data owner first encrypts the information before outsourcing the ciphertext to cloud A. The local backup is then deleted after he has checked the storage result. The owner of the data may switch cloud storage providers in the future and move certain data from cloud A to cloud B. The owner of the data then wishes to review the transfer's outcome. Finally, following a successful data transfer, the data owner requests that cloud A delete the data that was sent and verify the deletion outcome.

3. The concrete scheme

The following six algorithms are part of our new proposed scheme.

1) First, initialization

Create ECDSA public-private key pairs for the data owner, clouds A and B, and (PK_O, SK_O), (PK_A, SK_A), and (PK_B, SK_B), respectively. Then the data owner selects *k* secure hash functions, denoted by g_i: [1, N]$\rightarrow$ [1, m], that all map any integer in [1, N] to different cells in CBF. The data owner also selects a special tag tag_f for the file that will be uploaded to cloud A.

2) Data encryption

The data owner encrypts the outsourced file before uploading using a strong encryption mechanism to safeguard the confidentiality of the data.

i) After calculating the encryption key k = $H(tag_f \| SK_O)$ and encrypting the file C = $Enc_k(F)$ using k, the data owner decrypts the file using the IND-CPA safe encryption procedure. In order to ensure that the CBF won't be null after data transfer and deletion, the data owner divides the ciphertext *C* into n' blocks and simultaneously inserts $n - n'$ random blocks into the n' blocks at random points. The data owner then enters these arbitrary positions in a table *P F*.

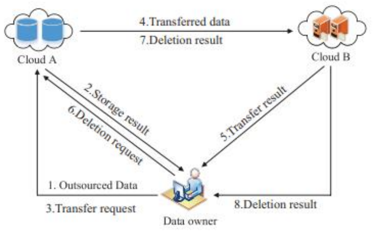

Fig. 4. The main processes of our scheme

ii) The data owner randomly selects a unique integer a_i as the index of each data block C_i, and then computes the hash values $H_i = H(tag_f \| a_i \| C_i)$. The outsourced data set is thus represented by the notation $D = ((a_1, C_1), ,(a_n, C_n))$. Finally, the data owner sends D and the file tag tag_f to cloud A.

3) Outsourcing data

A creates storage proof and stores *D* on the cloud.

The data owner then examines the storage outcome and removes the local backup.

i) After receiving data set *D* and file tag tag_f, cloud A saves *D* and builds a counting Bloom filter CBFs using the indexes (a_1, a_2,.., a_n), where i = 1, 2, , and n. tag_f is kept in cloud A as *D*'s index in the meantime. The proof $\lambda = (CBF_s, T_s, sig_s)$ is then sent to the data owner by cloud A, where *Sign* is an *ECDSA* signature algorithm and T_s is a timestamp. The cloud A computes a signature using the formula $sig_s = Sign_{SKA} (storage \| tag_f \| CBF_s \| T_s)$ before sending it.

ii) The data owner verifies the veracity of the storage proof after receiving it. In particular, the data owner first verifies the authenticity of the signatures. In the absence of valid sigs, the data owner outputs failure and chooses at random half of the indexes from (a_1, a_2,.., a_n) to verify the accuracy of the CBF_s.The data owner exits and reports failure if the CBF_s are incorrect; otherwise, the data owner deletes the local backup.

4) Data transfer

The data owner migrates some data blocks, or possibly the entire file, from cloud A to cloud B when he wants to switch service providers.

i) The data owner creates the index set of block indices, which identifies the data blocks that require migrating, first. A signature is then generated by the data owner using the formula $sig_t = Sign_{SKo} (transfer \| tag_f \| \phi \| T_t)$, where T_t is a timestamp. In the following step, the data owner creates a transfer request $Rt = (transfer, tag_f, \phi, Tt, sig_t)$, after which it is sent to cloud A. The data owner is also transmitting the hash values $\{H_i\}_{i \in \phi}$ and "i" to cloud B.

ii)The cloud A verifies the legitimacy of the transfer request R_t after receiving it. The cloud A computes a signature $sig_{ta} = Sign_{SKA} (R_t \| T_t)$, delivers the data blocks (ai, Ci)i to the cloud B, along with the signature sig_{ta}, and outputs failure if R_t is invalid. If R_t is valid, the cloud A continues and outputs success.

5) Transfer check

transfer and gives the data owner the transfer outcome.

i) The cloud B first verifies the accuracy of the R_t for the transfer and sig_{ta} for the signer. The cloud B checks that the equation $H_i = H(tag_f \| a_i \| m_i)$ holds, where $i \in \phi$, and if either of these is not true, the cloud B exits and outputs failure. If both of them are true, the cloud B continues. The cloud B requests that the cloud A send (ai, Ci) once more if $H_i \neq H(tag_f \| a_i \| C_i)$; otherwise, the cloud B moves on to Step ii).

ii)The cloud B stores the blocks $\{(a_i, C_i)\}_{i \in \phi}$ and creates a new counting Bloom filter (CBF_b) using the indexes

$\{a_i\}_{i \in \phi}$. The cloud B then generates a signature using the formula $sig_{tb} = Sign_{SKB}(success||tag_f || \phi ||T_t||CBF_b)$. The transfer evidence $\pi = (sig_{ta}, sig_{tb}, CBF_b)$ is then returned to the data owner by cloud B.

iii) The data owner examines the transfer outcome after receiving the signal. The data owner specifically verifies the authenticity of the signature sig_{tb}. To check the accuracy of the counting Bloom filter CBF_b, the data owner randomly selects half of the indices from the collection. If and only if all checks are successful, the data owner believes the transfer proof is accurate, and cloud B honestly stores the transferred data

6) Data deletion

When some data blocks have successfully been transferred to cloud B, the data owner may demand that cloud A delete those blocks.

i) To begin, the owner of the data creates a signature using the formula $sig_d = Sign_{SKA}(delete||tag_f ||\phi||T_d)$, where T_d is a timestamp. The data owner then creates and delivers to cloud A the data deletion request

$R_d = (delete, tagf, \phi, T_d, sig_d)$.

ii) The cloud A verifies Rd after receiving R_d. If R_d is invalid, cloud A terminates and reports failure; if not, cloud A overwrites the data blocks "(a_i, C_i)"$_{i \in \phi}$ and deletes them. In the meantime, cloud A gets a new counting Bloom filter CBF_d and removes indexes $\{aq\}_{q \in \phi}$ from the CBF_s. The cloud A then computes a signature

$sig_{da} = Sign(delete||Rd||CBF_d)$, and gives the data owner the deletion evidence = (sig_{da}, CBF_d).

iii) The data owner examines the signature sig_{da} after obtaining. If sig_{da} is false, the data owner exits and outputs failure; if true, the data owner randomly selects half of the indexes from and checks that $CBF(a_q) = 0$ and determines whether or not aq is a member of the CBF_d.The data owner believes that is true if the equations hold true.

Comment 1 It is important to keep in mind that the equality CBF(aq) = 0 holds indicates that at least one equation $h_i(a_q) = 0$ holds for $q \epsilon \psi$, indicating that a_q does not belong to the counting Bloom filter CBF_d.

V. SECURITY ANALYSIS

1. Data confidentiality

Without the matching data decryption key, the adversary cannot obtain any plaintext data due to data secrecy. In our system, the file is encrypted using the IND-CPA secure AES algorithm by the data owner. In the meantime,

$k = H(tagf ||SK_O)$, where H is a secure hash function and SK_O is the private key that is kept secret, is used to calculate the data decryption key.As a result, the attacker is unable to successfully generate a data decryption key. The key used to decode the data is also kept a secret by the data owner. This means that no opponent can gain the decryption key to obtain the plaintext data in the future.

2. Data integrity

The data integrity means that the transferred data must be intact, or the cloud B refuses to accept the data. Upon receiving the transferred data (a_i , C_i) from the cloud A and the hash values H_i from the data owner, the cloud B checks the equation

$H_i = H(tag_f ||a_i ||C_i)$, where $i \in \phi$. Note that $\{Hi\}_{i \in \phi}$ are computed by the data owner with a secure hash function. As a result, the cloud A and other enemies are unable to create a new data block (a_i, C'_i) that would satisfy the equation $H_i = H(tag_f||a_i ||C'_i)$.

That is, the cloud B can identify these malicious behaviours and will not accept the received data if the data is not honestly migrated from cloud A to cloud B or if the transferred data blocks are altered by the attackers during the migration process. Consequently, the data's integrity during transfer is ensured.

3. Public verifiability

It examine the transfer result's and the deletion result's verifiability, respectively.

The transfer result can be verified by the verifier using the transfer proof π and transfer request R_T. In particular, R_t's validity is checked by the verifier first. If R_t is accurate, it indicates that the data owner did request a data migration to cloud B. The authenticity of the signatures sig_{ta} and sig_{tb} is then further verified by the verifier. Keep in mind that cloud B won't purposefully work with cloud A to deceive the data owner. Therefore, if and only if both signatures are legitimate, the verifier can trust the transferred result. Additionally, the verifier validates the counting Bloom filter CBF_b to see if cloud B maintains the transferred data honestly.

Furthermore, the deletion result can be verified by the verifier who is the owner of the deletion evidence and the deletion request R_d. The verifier first determines whether Rd. If Rd is invalid, the data owner never needed to remove the data; if not, the verifier further confirms the reliability of the counting Bloom filter CBF_d and the validity of the signature sig_{da}. The verifier thinks the deletion evidence is true if and only if all the verifications succeed. It should be noted that the verifier does not require any private information for the transfer and deletion outcomes verification operations. In other words, our plan can fulfil public verifiability.

Although the verification could fail in the deletion phase due to a counting Bloom filter false positive, the likelihood of failure can be decreased. The likelihood of a false positive is given by $P_f = (1-e^{-kn/m})^k$ in Ref. [31]. The probability P_f achieves its minimal value, roughly $(0.6185)^{m/n}$, when $k = \ln 2 * (m/n)$. We fix $k = 20$ and $m/n = 29$ in our scheme. As a result, the probability P_f is almost equal to 2^{-20}, which is so minimal that it might as well be zero. As a result, we believe the verifier can effectively validate the deletion result every time.

IV. Efficiency Evaluation

1. Comparison

This section contains a theoretical comparison between our scheme and two earlier schemes[26,32]. One can get the following conclusions from Table 1.First off, each of these three methods can successfully delete data in a verifiable manner. Second, unlike the method in Ref.[32], our scheme and the strategy from Ref.[26] both enable verifiable data transfer while also enabling the integrity of the transferred data to be checked on a new cloud. Last but not least, unlike the system in Ref.[32], which calls for the introduction of a TTP, neither our scheme nor the scheme in Ref.[26] contain any TTP.

The results of our analysis of the theoretical performance are shown as time complexities in Table 2, where the symbols $\mathcal{E}, S, V, Exp, H$, and P stand for data encryption, signature generation, signature verification, exponentiation in G_1, hash computation, and pairing calculation, respectively. Additionally, n and l, correspondingly, are the sum of the data blocks and the total number of added/removed data blocks. We don't factor in additional calculations, such multiplication and addition, for the sake of simplicity.

Table 1. Functionality comparison

	Scheme[32]	Scheme[26]	Our scheme
TTP	✓	×	×
Data integrity	×	✓	✓
Provable transfer	×	✓	✓
Verifiable deletion	✓	✓	✓

Table 2. Complexity comparison

	Scheme[32]	Scheme[26]	Our scheme
Encrypt	$2\mathcal{E} + 4\mathcal{H}$	$n\mathcal{H} + 1\mathcal{E}$	$1\mathcal{H} + 1\mathcal{E}$
Storage	-	$n^2E + 2n\mathcal{P}$	$1\mathcal{S} + 1\mathcal{V} + 30n\mathcal{H}$
Transfer	-	$2\mathcal{S} + 2\mathcal{V} + 2l\mathcal{P} + (l+n)\mathcal{E}xp$	$3\mathcal{S} + 3\mathcal{V} + 31l\mathcal{H}$
Deletion	$1\mathcal{S} + 1\mathcal{V}$	$(l+1)(\mathcal{S} + \mathcal{V}) + (l+n)\mathcal{E}xp$	$2\mathcal{S} + 2\mathcal{V} + 30l\mathcal{H}$

2. Simulation results

It simulate our scheme and earlier schemes[26,32] using the PBC and OpenSSL libraries on the same Linux computer that has an Intel(R) Core(TM) i5-4590 processor clocked at 3.30GHz and 4 GB of main memory. In addition, we consider that k = 20, m/n = 29.The file size is increased from 1 megabyte (MB) to 8 megabytes (MB) in steps of 1 MB during the encryption phase, and the number of data blocks is set at 8000. The time and cost comparison is displayed in Figure 5. We can observe that the size of the encrypted data causes the time costs of the three techniques to rise. However, compared to the system in Ref.[32] and roughly on par with the plan in Ref.[26], our scheme's growth rate is considerably slower. Due to the fact that the techniques in Ref.[26] and Ref.[32] require significantly more hash calculations to create encryption keys and the MAC, respectively, the time cost of our scheme is lower than that of the other two schemes. Therefore, we believe that our method of file encryption is more effective.

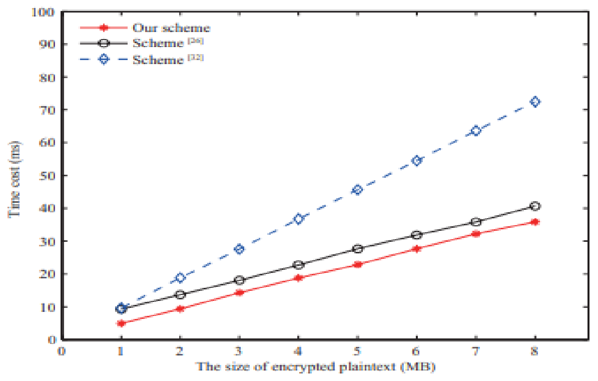

fig. 5. The time cost of data encryption

The computing cost in the storage phase is caused by the creation of storage proofs and the verification of storage results. The time required to generate a storage evidence is shown in Fig. 6. We discover that the time of our plan is substantially longer.The growth rate of Yang et al.'s scheme of Ref.[26] is comparatively larger than that of our system, while it is lower than that of Yang et al.'s plan of Ref.[26]. As a result, our method is more effective in producing the storage proof. The data owner then verifies the storage outcome, and Fig. 7 displays the performance comparison. Because our scheme only needs to perform a few hash calculations and a signature verification operation, we can see that its overhead is significantly lower than that of Yang et al.'s scheme in Ref.[26]. However, Yang et al.'s plan in Ref.[26] requires performing numerous bilinear pairing calculations.

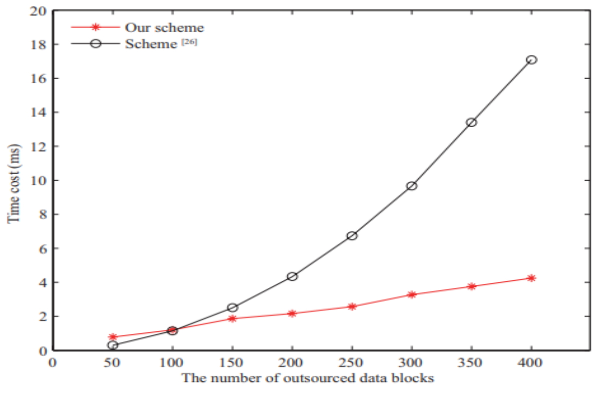

Fig. 6. The time of storage proof generation

We incrementally up the number of sent data blocks from 10 to 80 in order to simulate the data transfer. As shown in Fig. 8, we fix n = 400 and disregard the communication overhead for simplicity. The time expense rises as more data blocks are sent. Furthermore, Yang et al.'s method from Ref.[26] takes much longer because it needs to perform numerous bilinear pairing calculations to check the accuracy of the data. Only a few hash values must be calculated for our scheme. The hash computation is more effective than the bilinear pairing calculation, as you should be aware of. Our plan is therefore more effective.

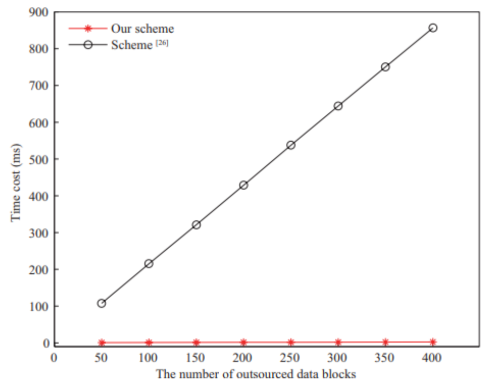

Fig. 7. The time of storage result verification

Finally, the data owner requests that the transferred data be deleted from cloud A. Lets fix n = 400, and the performance assessment is then shown in Fig. 9. Hao et al.'s plan from Ref.[32] has a nearly constant time overhead. The growth rate of Yang et al.'s scheme of Ref.[26] is comparatively higher, but the time cost of both our scheme and Yang et al.'s system of Ref.[26] will rise as more data blocks are destroyed. When there are more than 20 erased data blocks, Yang et al.'s method of Ref.[26] takes substantially longer. Therefore, one can believe that our plan is more effective at deleting the transferred data blocks.

.

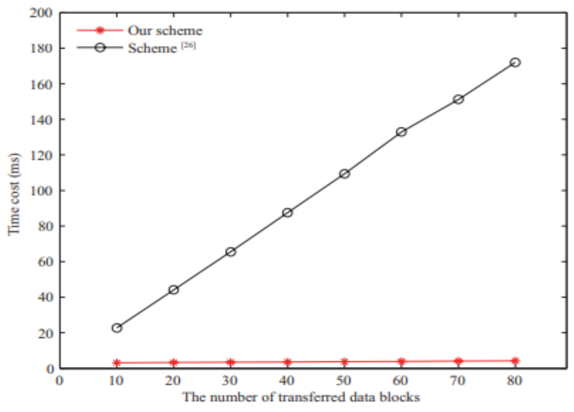

Fig. 8. The time cost of data transfer

V. Conclusions and Future Work

Conclusions. The data owner in cloud storage doesn't think the cloud server will honestly carry out the data transfer and delete procedures. We suggest a CBF-based secure data transfer mechanism that can also provide verifiable data erasure to address this issue.In our plan, cloud B can examine the transferred data. data integrity, which can ensure that all of the data has been moved. Additionally, the cloud A should use CBF to produce a deletion evidence after deletion, which the data owner will use to confirm the deletion outcome. As a result, cloud A is unable to act intentionally and deceive the data owner. Finally, the outcomes of the simulation and security analysis verify the viability and security of our proposal, respectively.

Future work

This strategy takes into account the data movement between two different cloud servers, just like all other solutions already in place. However, as cloud storage technology has advanced, the data owner may want to migrate the outsourced data from one cloud to two or more target clouds at once. However, the multi-target clouds might band together to maliciously defraud the data owner. Therefore, we need to continue investigating the verifiable data migration across three or more clouds.

REFERENCES

[1] C. Yang and J. Ye, “Secure and efficient fine-grained data access control scheme in cloud computing”, Journal of High Speed Networks, Vol.21, No.4, pp.259–271, 2015.
[2] X. Chen, J. Li, J. Ma, et al., “New algorithms for secure outsourcing of modular exponentiations”, IEEE Transactions on Parallel and Distributed Systems, Vol.25, No.9, pp.2386–2396, 2014.
[3] P. Li, J. Li, Z. Huang, et al., “Privacy-preserving outsourced classification in cloud computing”, Cluster Computing, Vol.21, No.1, pp.277–286, 2018.
[4] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions”, Future Generation Computer Systems, Vol.79, pp.849–861, 2018.
[5] W. Shen, J. Qin, J. Yu, et al., “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, 2019.
[6] R. Kaur, I. Chana and J. Bhattacharya J, “Data deduplication techniques for efficient cloud storage management: A systematic review”, The Journal of Supercomputing, Vol.74, No.5, pp.2035–2085, 2018.
[7] Cisco, “Cisco global cloud index: Forecast and methodology, 2014–2019”, available at: https://www.cisco.com/c/en/us-/solutions/collateral/service-provider/global-cloud-index-gci/ white-paper-c11-738085.pdf, 2019-5-5.

[8] Cloudsfer, “Migrate & backup your files from any cloud to any cloud”, available at: https://www.cloudsfer.com/, 2019-5-5. [9] Y. Liu, S. Xiao, H. Wang, et al., “New provable data transfer from provable data possession and deletion for secure cloud storage”, International Journal of Distributed Sensor Networks, Vol.15, No.4, pp.1–12, 2019.

[10] Y. Wang, X. Tao, J. Ni, et al., “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.

[11] Y. Luo, M. Xu, S. Fu, et al., “Enabling assured deletion in the cloud storage by overwriting”, Proc. of the 4th ACM International Workshop on Security in Cloud Computing, Xi’an, China, pp.17–23, 2016.

[12] C. Yang and X. Tao, “New publicly verifiable cloud data deletion scheme with efficient tracking”, Proc. of the 2th International Conference on Security with Intelligent Computing and Big-data Services, Guilin, China, pp.359–372, 2018.

[13] Y. Tang, P.P Lee, J.C. Lui, et al., “Secure overlay cloud storage with access control and assured deletion”, IEEE Transactions on Dependable and Secure Computing, Vol.9, No.6, pp.903–916, 2012.

[14] Y. Tang, P.P.C. Lee, J.C.S. Lui, et al., “FADE: Secure overlay cloud storage with file assured deletion”, Proc. of the 6th International Conference on Security and Privacy in Communication Systems, Springer, pp.380-397, 2010.

[15] Z. Mo, Y. Qiao and S. Chen, “Two-party fine-grained assured deletion of outsourced data in cloud systems”, Proc. of the 34th International Conference on Distributed Computing Systems, Madrid, Spain, pp.308–317, 2014.

[16] M. Paul and A. Saxena, “Proof of erasability for ensuring comprehensive data deletion in cloud computing”, Proc. of the International Conference on Network Security and Applications, Chennai, India, pp.340–348, 2010.

[17] A. Rahumed, H.C.H. Chen, Y. Tang, et al., “A secure cloud backup system with assured deletion and version control”, Proc. of the 40th International Conference on Parallel Processing Workshops, Taipei City, Taiwan, pp.160–167, 2011.

[18] B. Hall and M. Govindarasu, “An assured deletion technique for cloud-based IoT”, Proc. of the 27th International Conference on Computer Communication and Networks, Hangzhou, China, pp.1–8, 2018.

[19] L. Xue, Y. Yu, Y. Li, et al., “Efficient attributebased encryption with attribute revocation for assured data deletion”, Information Sciences, Vol.479, pp.640–650, 2019.

[20] L. Du, Z. Zhang, S. Tan, et al., “An Associated Deletion Scheme for Multi-copy in Cloud Storage”, Proc. of the 18th International Conference on Algorithms and Architectures for Parallel Processing, Guangzhou, China, pp.511–526, 2018.

[21] C. Yang, X. Chen and Y. Xiang, “Blockchain-based publicly verifiable data deletion scheme for cloud storage”, Journal of Network and Computer Applications, Vol.103, pp.185–193, 2018.

[22] Y. Yu, J. Ni, W. Wu, et al., “Provable data possession supporting secure data transfer for cloud storage”, Proc. of 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications(BWCCA 2015), Krakow, Poland, pp.38–42, 2015.

[23] J. Ni, X. Lin, K. Zhang, et al., “Secure outsourced data transfer with integrity verification in cloud storage”, Proc. of 2016 IEEE/CIC International Conference on Communications in China, Chengdu, China, pp.1–6, 2016.

[24] L. Xue, J. Ni, Y. Li, et al., “Provable data transfer from provable data possession and deletion in cloud storage”, Computer Standards & Interfaces, Vol.54, pp.46–54, 2017.

[25] Y. Liu, X. Wang, Y. Cao, et al., “Improved provable data transfer from provable data possession and deletion in cloud storage”, Proc. of Conference on Intelligent Networking and Collaborative Systems, Bratislava, Slovakia, pp.445–452, 2018.

[26] C. Yang, J. Wang, X. Tao, et al., “Publicly verifiable data transfer and deletion scheme for cloud storage”, Proc. of the 20th International Conference on Information and Communications Security (ICICS 2018), Lille, France, pp.445–458, 2018.

[27] B.H. Bloom, “Space/time trade-offs in hash coding with allowable errors”, Communications of the ACM, Vol.13, No.7, pp.422–426, 1970.

[28] A. Broder and M. Mitzenmacher, “Network applications of bloom filters: A survey”, Internet Mathematics, Vol.1, No.4, pp.485–509, 2004.

[29] J. Wang, X. Chen, X. Huang, et al., “Verifiable auditing for outsourced database in cloud computing”, IEEE transactions on computers, Vol.64, No.11, pp.3293–3303, 2015

[30] L. Fan, P. Cao, J. Almeida, et al., “Summary cache: A scalable wide-area web cache sharing protocol”, IEEE/ACM Transactions on Networking, Vol.8, No.3, pp.281–293, 2000.

[31] O. Rottenstreich, Y. Kanizo and I. Keslassy, “The variableincrement counting Bloom filter”, IEEE/ACM Transactions on Networking, Vol.22, No.4, pp.1092–1105, 2014.