A STUDY ON INTEGRITY VERIFICATION SCHEME OF COMPLETENESS AND ZERO-KNOWLEDGE FOR MULTI-CLOUD STORAGE

Dr. M. CHANDRA SEKHAR, Associate Professor, Department of CSE, Nalla Malla Reddy Engg College, Hyderabad.

DEEPU JIMMYJOEL LAZARUS, Research Scholar, Associate Professor, Department of Computer Science & Technology, Sri Krishnadevaraya University, Anantapur.

Dr.G.A.RAMACHANDRA, Associate Professor, Department of Computer Science & Technology, Sri Krishnadevaraya University, Anantapur.

ABSTRACT:

In order to resist the attacks from the malicious cloud service providers (CSPs) and organizer, an integrity verification scheme of completeness and zero-knowledge for multi-cloud is proposed. Firstly, we adopt an interactive proof of system to verify the integrity of customer data. Secondly, the change of file blocks is recorded and the hash value of each block is generated through the index-hash table (IHT) in the verification process. Finally, the hash value of each block is updated through this IHT to support dynamic updates to user's data. Compared with the original scheme, this scheme can resist the malicious attacks and reduce the computation communication costs; it can realize the dynamic update of data and has the properties of completeness, reliability and zero knowledge. Experimental results also show that this scheme can resist the malicious attacks and reduce the computation communication costs. Provable data possession (PDP) is a technique for ensuring the integrity of data in storage outsourcing. In this paper, we address the construction of an efficient PDP scheme for distributed cloud storage to support the scalability of service and data migration, in which we consider the existence of multiple cloud service providers to cooperatively store and maintain the clients' data. We present a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy. We prove the security of our scheme based on multiprover zero-knowledge proof system, which can satisfy completeness, knowledge soundness, and zero-knowledge properties. In addition, we articulate performance optimization mechanisms for our scheme, and in particular present an efficient method for selecting optimal parameter values to minimize the computation costs of clients and storage service providers. Our experiments show that our solution introduces lower computation and communication overheads in comparison with noncooperative approaches.

Keywords: *Provable data possession, integrity verification, dynamic operations, zero-knowledge, cloud storage security.*

1. INTRODUCTION

Cloud computing as a new computing model, is generally considered to be another technological change and IT industry growth point after the Internet [1]. Hybrid cloud model is the future development trend of cloud computing, it is one of the cloud deployment modes, which is composed of at least one private cloud and one public cloud. Within a multi-cloud, an organization can offer and manage in-house and out-house resources. Moreover, the multi-cloud contributes to balance the load and automatically redeploy processing logic in the event of failures. However, the cloud service providers (CSPs) are usually not trustworthy. They may conceal the data loss or error from the users for their own benefit. Even more, they might delete rarely accessed user data for saving storage space. As a result, many users are still hesitant to use cloud storage due to security and confidentiality threats toward their outsourced data. Hence, a method to verify the integrity of the user data in multi-cloud is urgent to be put forward.

2. Related Work

Zhu et al.[2, 3] propose a cooperative provable data possession (CPDP) scheme for multi-cloud storage based on the techniques of fragment structure. The verification protocol of this scheme is

similar to Schnorr [4]. This scheme utilizes the techniques of fragment structure, hash index hierarchy (HIH), and homomorphic verifiable response (HVR). In their scheme, a set of public verification information is stored in trusted third party (TTP). One of CSPs is treated as the organizer, who is responsible for communication with the verifier. Later, by issuing a challenge to the organizer, the user (verifier) can check the integrity and availability of outsourced data with public information stored in TTP. Their scheme can combine responses from multiple CSPs into a single final response since the responses are homomorphic. However, the scheme has a flaw [5], which the malicious CSPs or organizer can pass validation without storing users' data.

Wang Huaqun [7] proposes an identity-based distributed provable data possession (DPDP) in multi-cloud storage, the security of the scheme is used to solve the problem by computation Diffie-Hellman (CDH). In addition, the scheme also supports private verification, authentication, authorization and based on public verification, but it does not support the dynamic update of data.

This paper proposes a integrity verification scheme of completeness and zero-knowledge for multi-cloud, which meets dynamic data updates and knowledge soundness. To achieve dynamic data updates, we introduce the index-hash table (IHT) construction. A scheme to support fully dynamic operations based on the improved IHT is brought forward.

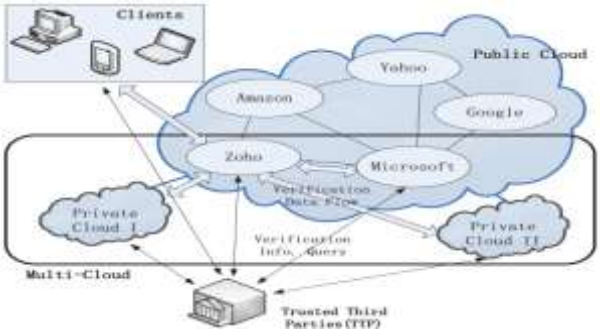

Figure 1 : Verification Architecture for Data Integrity.

In this architecture, we consider the existence of multiple CSPs to cooperatively store and maintain the clients' data. Moreover, a cooperative PDP is used to verify the integrity and availability of their stored data in all CSPs. The verification procedure is described as follows: first, a client (data owner) uses the secret key to preprocess a file which consists of a collection of *n* blocks, generates a set of public verification information that is stored in TTP, transmits the file and some verification tags to CSPs, and may delete its local copy. Then, by using a verification protocol, the clients can issue a challenge for one CSP to check the integrity and availability of outsourced data with respect to public information stored in TTP.

We neither assume that CSP is trust to guarantee the security of the stored data, nor assume that data owner has the ability to collect the evidence of the CSP's fault after errors have been found. To achieve this goal, a TTP server is constructed as a core trust base on the cloud for the sake of security. We assume the TTP is reliable and independent through the following functions [12]: to setup and maintain the CPDP cryptosystem; to generate and store data owner's public key; and to store the public parameters used to execute the verification protocol in the CPDP scheme. Note that the TTP is not directly involved in the CPDP scheme in order to reduce the complexity of cryptosystem.

The rest of the paper is organized as follows. Problem Statement is shown in Section 3. Section 4 describes Architecture and Techniques. Section 5 describes Our Scheme. Experiments are shown in Section

6. Section 7 summarizes our results and gives the conclusions of our research.

3.Problem Statement

3.1. The Malicious CSPs Deceives the Clients

The malicious CSPs can deceive the clients by performing the procedures in the following.

Support P_k has deleted all the clients' remote data. It can generate the valid response to the organizer O as follows: P_k accesses TTP and gets the corresponding verification parameters $\{\chi_i\}$. Then, it picks random $\sigma'_k \in G_T$, $\mu_{j,k} \in \mathbb{Z}_p$, $j \in [1,s]$, $r_k \in \mathbb{Z}_p$ and calculates

$$\pi_k = e\left(\prod_{(i,v_i)\in Q} H_{\xi_k}^{(2)}(\chi_i)^{v_i}, H_1\right) \cdot e\left(\prod_{j\in[1,s]} u_j^{\mu_{j,k}}, H_2\right) \cdot e(S^{-r_k}\sigma'_k, h)^{-1} \quad (1)$$

$$\eta_k = g^{r_k} \quad (2)$$

The malicious CSPs get $\theta_k = (\pi_k, \sigma'_k, \mu_k, \eta_k)$ and send θ_k to the organizer O.

3.2. The Malicious Organizer Deceives the Clients

Suppose the organizer O is malicious. It can generate the valid response without the stored clients' remote data as follows.

After getting the challenge $Q = \{(i, v_i)\}$, the organizer O can accesses TTP and get the corresponding verification parameters $\{\chi_i\}$. Then, it picks random $\sigma'_k \in G$, $\mu' = \{\mu'_j\} \in \mathbb{Z}_p^s$ and calculates

$$\pi' = e\left(\prod_{(i,v_i)\in Q} H_{\xi_k^{(2)}}(\chi_i)^{v_i}, H'_1\right) \cdot e\left(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2\right) \cdot e(\sigma', h)^{-1}$$

Then, the organizer O sends $\theta = (\pi', \sigma', \mu')$ to the verifier V. It is easy to see that θ can pass the verifier's checking.

According to the forgery procedure, the following equation holds

$$\pi' \cdot e(\sigma', h) = e\left(\prod_{(i,v_i)\in Q} H_{\xi_k^{(2)}}(\chi_i)^{v_i}, H'_1\right) \cdot e\left(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2\right) \quad (3)$$

4.Architecture and Techniques

4.1. The Architecture of System

A multi-cloud storage involves three different entities:

- *The cloud user*. Who has a huge number of data files to be stored in multiple clouds and is entitled to access and manipulate stored data.
- *CSPs*. Who work in coordination to provide data storage service and have enough storage space and significant computation resources?
- *TTP*. Who has expertise and capabilities that the cloud users do not have. It is trusted to evaluate the reliability of the cloud storage service for the user.

The verification procedure is described as follows: first, a client (data owner) uses the secret key to preprocess a file which consists of a collection of n blocks, generates a set of public verification information that is stored in TTP, transmits the file and some verification tags to CSPs, and may delete its local copy. Then, by using a verification protocol, the clients can issue a challenge for one CSPs to check the integrity and availability of outsourced data with respect to public information stored in TTP.

4.2. Hash Index Hierarchy

Hash index hierarchy (HIH)[2] shows the relationship of user's data. In order to make our scheme support dynamic data operations, we increase a new column C_i to record the

serial number of CSPs in the index table χ. Given a collision-resistant hash function $H_k(\cdot)$, we can construct a HIH function $\mathcal{H}$ below:

(1) *Express layer*: Input s random $\{\tau_i\}_{i=1}^{s}$ and the file name F_n, computes $\xi^{(1)} = H_{\sum_{i=1}^{s}\tau_i}(F_n)$, offers the abstract representation of the stored resources.

(2) *Service layer*: Input $\xi^{(1)}$ and the cloud name C_k, computes $\xi_k^{(2)} = H_{\xi^{(1)}}(C_k)$, offers and manages cloud storage services.

(3) *Storage layer*: Input $\xi_k^{(2)}$, a block number i, and its index record $\chi_i = B_i \| V_i \| R_i \| C_i$,computes $\xi_{i,k}^{(3)} = H_{\xi_k^{(2)}}(\chi_i)$, where B_i is the sequence number of a block, V_i is the version number, and R_i is a random integer, realizes data storage on many physical devices.

4.3. Index-Hash Table

To adapt to multi-cloud storage, the index table χ in the CPDP scheme needs to increase a new column C_i to record the serial number of CSPs, which stores the i-th block. By using this structure, it is obvious that our scheme can also support dynamic data operations in multi-cloud storage (as shown in Table 1).

Table 1. IHT with Random Values.

C_i	B_i	V_i	R_i
0	0	0	0
1	1	2	r_1'
2	2	1	r_2
⋮	⋮	⋮	⋮
n	n	1	r_n

5. Our Scheme

5.1. The verification

In our scheme, we split a file into n blocks, each of block has a sector which is notated as *s*, so a file *F* can be expressed as the file with $n \times s$ section, $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$. We use *t* to describe the private key of the organizer CSPs, use *c* to express the number of a clouds store file. σ describes the set of tags, $\sigma = \{\sigma_i\}_{i\in[1,n]}$. *Q* describes the set of index-coefficient pairs, $Q = \{i, v_i\}$. θ describes the response for the challenge *Q* (as shown in Table 2).

Table 2. The Signal and its Explanation.

Notations	Descriptions
n	The number of blocks in a file
s	The number of sectors in each block
t	The private key of the organizer CSPs
c	The number of clouds to store a file
F	The file with $n \times s$ section, F=$\{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$
σ	The set of tags , σ = $\{\sigma_i\}_{i\in[1,n]}$
Q	The set of index-coefficient pairs, Q= $\{i, v_i\}$
θ	The response for the challenge Q

We use the layered Hash index and the authentication response technology to verify the integrity of customer, it is shown as follows:

$KeyGen(1^k)$: Take a security parameter *k* as input, and return a key *sk* or a public-secret keypair (pk, sk).

$TagGen\ (sk, F, \mathcal{CP})$: Take as inputs a secret key *k*, a file *F*, and a set of cloud storage providers $\mathcal{CP} = \{CP_k\}$, and returns the triples (ξ, ψ, σ), where ξ is the secret in tags, $\psi = (u, \mathcal{H})$ is a set of verification parameters *u* and index hierarchy $\mathcal{H}$ for *F*, $\sigma = \{\sigma^{(k)}\}_{CP_k\in\mathcal{CP}}$ denotes a set of all tags, $\sigma^{(k)}$ is the tag of the fraction $F^{(k)}$ of *F* in CP_k.

$Proof\ (\mathcal{CP}, Verifier)$: It is a protocol among the providers $\{CP_k\}_{k\in[1,c]}$, an organizer *O*, and a verifier *V* with the common input (pk, ψ). The protocol can be expressed as follows

(1) *Commitment* $(O \rightarrow V)$: The organizer picks a random $\gamma \in \mathbb{Z}_p$ and sends $H_1' = H_1^{\gamma}$ and $\pi' \leftarrow \left(\prod_{CP_k \in} \pi_k\right)^{\gamma}$ to the verifier.

(2) *Challenge 1* $(O \leftarrow V)$: The verifier picks challenged index-coefficient pairs $Q = \{(i, v_i)\}_{i \in I}$, and sends Q to the organizer, where $I \subseteq [1, n]$ and $v_i \in \mathbb{Z}_p^*$.

(3) *Challenge 2*$(\mathcal{CP} \leftarrow O)$: The organizer forwards $Q_k = \{(i, v_i)\}_{m_i \in CP_k} \subseteq Q$ to each in $CP_k \in \mathcal{CP}$.

(4) *Response 1* $(\mathcal{CP} \leftarrow O)$: In this stage, we suppose the organizer O is malicious. It can generate the valid response without the stored clients' remote data. In order to prevent the organizers obtained user data information, we increased an additional data block and the label to hide the location of the user data stored on every server of CSPs, so that each CSPs can return the response to the organizer, reach the organizers unable to obtain the user's information. User generated an additional data blocks $m_{m',j}$ and calculate the label of $m_{m',j}$ as

$$\sigma_{m'} = \left(\prod_{j=1}^{a} u_j^{m_{m',j}}\right)^{\beta} \quad (4)$$

CP_k picks random $r_k \in \mathbb{Z}_p$ and $\lambda_{j,k} \in \mathbb{Z}_p$, j $\in [1, s]$, and calculates the label and response information:

$$\sigma_k' = \sigma_{m'} \cdot \prod_{(i,v_i) \in Q_k} {\sigma_i}^{v_i}$$

$$\mu_{j,k} = \lambda_{j,k} + \sum_{(i,v_i) \in Q_k} v_i \cdot m_{i,j} + m_{m',j} \quad (5)$$

Let $\mu_k = \{\mu_{j,k}\}_{j \in [1,s]}$, CP_k sends $\theta_k = (\sigma_k', \mu_k)$ to O.

(5) *Response2* $(O \rightarrow V)$: The organizer receives all the responses $\{\theta_k\}$ from $\{CP_k\}_{k \in [1,c]}$, and calculates $\sigma' \leftarrow \left(\prod_{CP_k \in} \sigma_k'\right)^{\gamma}, \mu_j' \leftarrow \sum_{CP_k \in} \gamma \cdot \mu_{j,k}$

Let $\mu' = \{\mu_j'\}_j \in [1, s]$. Then, the organizer sends $\theta = (\sigma', \mu')$ to the verifier.

(6) *Verification*: The verifier computes whether Eq.(6) holds. If Eq.(6) holds, the data is integrity.

$$\pi' \cdot e(\sigma', h) = e\left(\prod_{(i,v_i) \in Q} H_{\xi_k^{(2)}}(\chi_i)^{v_i}, H_1'\right) \cdot e\left(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2\right) \quad (6)$$

5.2. Implementation of Dynamic Operation

In Cloud Computing, the outsourced data might not only be accessed but also be up-dated frequently by clients for various application purposes [8]. Based on the existing work [9, 10], we purpose a scheme to support fully dynamic operations including insertion $(\mathcal{I})$, deletion $(\mathcal{D})$ and modification $(\mathcal{M})$ for the cloud storage.

To adapt to multi-cloud storage, the index table χ in the CPDP scheme needs to increase a new column C_i to record the serial number of CSPs, which stores the i-th block. By using this structure, it is obvious that our CPDP scheme can also support dynamic data operations in multi-cloud storage.

The record in IHT is represented as $it_i = \{B_i, V_i, R_i, C_i, S_i\}$, where B_i is the original number of block, V_i stores the version number of updates for block, R_i is a random integer to avoid collision and S_i is a signature to avoid the cloud to forge this item. In addition, we use *SN* to denote the real number i of data blockm_i. During the Setup phase, except computing the verification metadata, the client still needs to build an IHT to record the real-time status of the outsourced data, the client sets B_i, V_i, R_i, C_i, and computes

$$\xi_{i,k}^{(3)} = H_{\xi_k^{(2)}}(\chi_i), \varrho_i = \left(H_{\xi_k^{(2)}}(\chi_i)\right)^{\iota} \quad (7)$$

Where ι is an up-dated key? Finally, the client sends the data file F, the verification meta-data ϕ and the IHT ζ to the cloud.

Data Modification: The client first sends the update request $\{\mathcal{M}, \mathrm{i}, \mathrm{C_i}\}$ to the multi-cloud storage. The cloud returns the item $\mathrm{it_i}$ from the IHT to the client. The client computes the hash value $\xi_{\mathrm{i,k}}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| V_i \| R_i || C_i)$, and verifies whether the item in table is intact with $e(\varrho_i, g) = e\left(\xi_{i,k}^{(3)}, \pi\right)$, where g is a generator of G, ι is an updated key, ρ is a public key, $\rho = g^{\iota}$. If the authentication fails, the verifier rejects by emitting false and halt. Next, the client modifies the version number by $V_i \leftarrow V_i + 1$, chooses a new signature random R_i to compute the new signature $\varrho_{i''}$ and the new hash $\xi_{i,k}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| V_i \| R_i || C_i)$, and calculates $\sigma_{i,k} \leftarrow \left(\xi_{i,k}^{(3)}\right)^{\alpha} \cdot \left(\prod_{j=1}^{s} u_j^{m_{i,j}}\right)^{\beta}$ by using $sk_u = (\alpha, \beta)$. Finally, client sends the updated data $update = \{m_i', \sigma_i', \chi_i'\}$ to the multi-cloud for update (as shown in Figure 1).

SN	B_i	C_i	V_i	R_i	S_i
1	1	1	1	232	ϱ_1
2	2	1	1	145	ϱ_2
3	3	2	1	432	ϱ_3
4	4	3	1	378	ϱ_4
5	5	2	1	512	ϱ_5

⇒

SN	B_i	C_i	V_i	R_i
1	1	1	1	232
2	2	1	1	145
3	3	2	2	432
4	4	3	1	378
5	5	2	1	512

Figure 1. Data Modification.

Data Insertion: The client first sends the update request $\{\mathcal{I}, i, C_i\}$ to the multi-cloud storage. The cloud returns the item it_i from the IHT to the client. The client computes the hash value $\xi_{i,k}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| V_i \| R_i || C_i)$, and verifies whether the item in table is intact with $e(\varrho_i, g) = e\left(\xi_{i,k}^{(3)}, \pi\right)$, where g is a generator of G, ι is an updated key, ρ is a public key, $\rho = g^{\iota}$. If the authentication fails, the verifier rejects by emitting false and halt. Next, the client modifies the version number by $V_i \leftarrow 1$, chooses a new signature random R_i, $B_i \leftarrow B_{i-1}$, $SN_i \leftarrow SN_{max} + 1$ to compute the new signature $\varrho_{i''}$ and the new hash $\xi_{i,k}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| V_i \| R_i || C_i)$, and calculates $\sigma_{i,k} \leftarrow \left(\xi_{i,k}^{(3)}\right)^{\alpha} \cdot \left(\prod_{j=1}^{s} u_j^{m_{i,j}}\right)^{\beta}$ by using $sk_u = (\alpha, \beta)$. Finally, client sends $\{m_i', \sigma_i', \chi_i'\}$ to the multi-cloud for update (as shown in Figure 2).

SN	B_i	C_i	V_i	R_i	S_i
1	1	1	1	232	ϱ_1
2	2	1	1	145	ϱ_2
3	3	2	2	432	$\varrho_{3'}$
4	4	3	1	378	ϱ_4
5	5	2	1	512	ϱ_5

⇒

SN	B_i	C_i	V_i	R_i	S_i
1	1	1	1	232	ϱ_1
2	2	1	1	145	ϱ_2
3	3	2	2	432	$\varrho_{3'}$
6	4	2	1	1024	$\varrho_{4'}$
4	5	3	1	378	$\varrho_{5'}$
5	6	2	1	512	$\varrho_{6'}$

Figure 2. Data Insertion.

Data Deletion: The client first sends the update request $\{\mathcal{D}, i, C_i\}$ to the multi-cloud storage. The cloud returns the item it_i from the IHT to the client. The client computes the hash value $\xi_{i,k}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| V_i \| R_i || C_i)$, and verifies whether the item in table is intact with $e(\varrho_i, g) = e\left(\xi_{i,k}^{(3)}, \pi\right)$, where g is a generator of G, ι is an updated key, ρ is a public key, $\rho = g^{\iota}$. If the authentication fails, the verifier rejects by emitting false and halt. Next, the client modifies the version number by $V_{i'} \leftarrow 0$, chooses a new signature random R_i to compute the new signature $\varrho_{i''}$ and the new hash $\xi_{i,k}^{(3)} = H_{\xi_k{}^{(2)}}(B_i \| 0 \| R_i || C_i)$, calculates $\sigma_{i,k} \leftarrow \left(\xi_{i,k}^{(3)}\right)^{\alpha}$ by using $sk_u = (\alpha, \beta)$, and delete i-th record to get a new ψ'. Finally, client sends $\{m_i', \sigma_i', \sigma_i\}$ to the multi-cloud for update(as shown in Figure 3).

SN	B_i	C_i	V_i	R_i	S_i
1	1	1	1	232	ϱ_1
2	2	1	1	145	ϱ_2
3	3	2	2	432	$\varrho_{3'}$
6	4	2	1	1024	$\varrho_{4'}$
4	5	3	1	378	$\varrho_{5'}$
5	6	2	1	512	$\varrho_{6'}$

⇒

SN	B_i	C_i	V_i	R_i	S_i
1	1	1	1	232	ϱ_i
2	2	1	1	145	ϱ_i
3	3	2	2	432	$\varrho_{3'}$
6	4	2	1	1024	$\varrho_{4'}$
4	5	3	1	378	$\varrho_{5'}$

Figure 3. Data Deletion.

6.Experimental Results

In this paper, two local IBM servers with two Intel Core 2 processors at 2.16 GHz and 500M RAM running Windows Server 2003

are used to simulate the integrity of data. A MNT curve is utilized in the experiment, with base field size of 160 bits and the embedding degree6. The security level is chosen to be 80bits, which means $|p| = 160$.

Different parameters such as sector number of per block s are used in our experiment. Our analysis shows that the value of s should grow with the increase of the size of file. So, our experiments are carried out to reduce computation and communication costs, as follows: the size of file from 10KB to 20MB is chosen; the sector stored files are changed from 20 to 250 in terms of file sizes; and the sampling ratios are changed from 10 to 50 percent. The experimental results are shown in Figure 4. Next, in order to validate the theoretical results, we change the sampling ration w from 10 to 50 percent for a 10 MB file and 250 sectors per block in a multi-cloud $\mathcal{CP} = \{CP_1, CP_2, CP_3\}$, in which the proportions of data blocks are 50, 30, and 20 percent in three CSPs, respectively. In the Figure 5, our experimental results show that the computation and communication costs of commitment and challenge are slightly changed along with the sampling ratio, but those for response and verification grow with the increase of the sampling ratio. Furthermore, the proportions of data blocks in each CSPs have greater influence on the computation costs of challenge and response processes. In summary, our scheme has better performance than noncooperative approach.

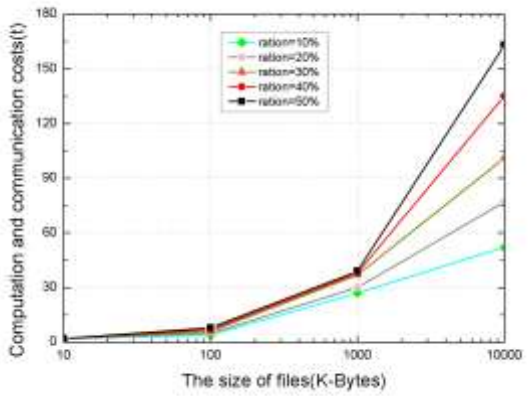

Figure 4. Experimental results under different file size, sampling radio, and sector number.

7.Conclusion

In this paper, according to HIH, HVR and IHT, we propose an integrity verification scheme of completeness and zero-knowledge for multi-cloud storage. Our scheme can resist the cloud storage service provider forgery attack. The security of our scheme is analysed, the results show that the program has the completeness, reliability, and security attributes, such as zero-knowledge, compared with the original plan, the paper reduces the computational and communication overhead, while supporting dynamic update operations data. Furthermore, we optimized the probabilistic query and periodic verification to improve the audit performance. Our experiments clearly demonstrated that our approaches only introduce a small amount of computation and communication overheads. Therefore, our solution can be treated as a new candidate for data integrity verification in outsourcing data storage systems.

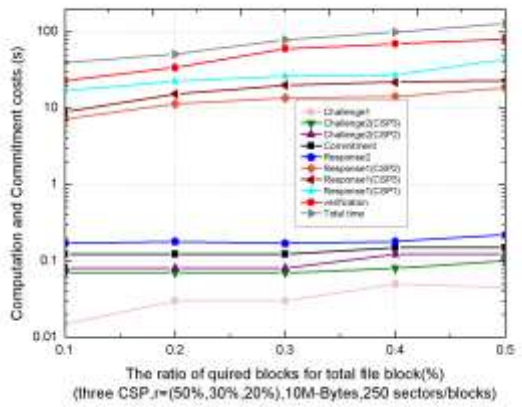

Figure 5. Experimental results under different file size, sampling ratio, and sector number.

REFERENCES

[1] Feng Guodeng, Zhang Yan, Zhangyan et al., "Research on security of cloud

computing," *Journal of software*, vol.22, no.1, pp.71-88, 2011.

[2] Zhu Yan, Han Yujing, Chen Shimin et al, "Collaborative Integrity Verification in Hybrid Clouds", *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 197-206, 2011.

[3] Zhu Yan, Hu Hongxin, Ahn Gail-Joon et al., "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.

[4] Shacham Hovav and Waters Brent, "Compact proofs of retrievability," *Joural of cryptology*, vol.26, no.3, pp.442-483, 2013.

[5] Wang Huaqun and Zhang Yuqing, "On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol.25, no.1, pp.264-267, 2014.

[6] Zhou Enguang, Li Zhoujun, Guo Hua et al, "An improved hybrid cloud environment that can prove the data hold program," *Journal of Tsinghua University* (NATURAL SCIENCE EDITION), vol.53, no.12, pp.1731-1736, 2013.

[7] Huaqun Wang, "Identity-Based Distributed Provable Data Possession in Multicloud Storage," *IEEE Transactions on Services Computing*, vol.8, no.2, pp.328-340, 2015.

[8] Wang Cong, Wang Qian, Ren Kui, et al., "Privacy-preserving public auditing for data storage security in cloud computing," *Proceedings-IEEE INFOCOM*, San Diego, United states, 2010.

[9] Barsoum A F, Hasan M A, "Integrity Veridication of Multiple Data Copies over Untrusted Cloud Servers," *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (CCGRID'12), Ottawa Canada. IEEE Computer Society Washington, DC, USA: IEEE, pp.829-834, 2012.

[10] Zhu Yan, Wang Huaixi, Hu Zexing et al, "Dynamic audit services for integrity verification of outsourced storages in clouds," *Proceedings of the ACM Symposium on Applied Computing*, Taichung, TaiWan. New York, NY USA: ACM, pp.1550-1557, 2011.