ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 3, March 2013

A Scalable Privacy-Preserving Verification Correctness Protocol to Identify Corrupted Data in Cloud Storage

Neethu Mariam Joseph, Esther Daniel, Vasanthi N A

***Abstract*— Cloud computing prevails over the whole world which makes the mankind to rely more on a number of online storage systems to back up data or for using it in real time which gives an anywhere, anytime access. Thus lot of sensitive data will be stored into the cloud. As the user's data are stored and maintained out of user's premises, these services bring with it, concerns of security vulnerabilities for all the services provided by them. There is a lot of research being done to point out the issues with the service providers and cloud security in general. This paper provides a privacy-preserving data integrity protection by enabling public auditability for cloud storage and implements a scalable framework that addresses the construction of an interactive audit protocol to prevent the fraudulence of prover and the leakage of verified data in cloud storage by reducing the overhead in computation, communication and storage. Thus it provides a lightweight, efficient and privacy –preserving auditing scheme which will also identifies the corrupted data. The framework is based on an interactive PDP protocol that uses the challenge – response algorithms and a verification protocol. The protocol supports dynamic operation that allows the granted users to perform insertion, deletion and updation. The protocol allows a Third party Auditor who works on behalf of the Data owner who has a large amount of data to be stored in the cloud, to manage or monitor outsourced data. The experimental results show that the protocol is efficient, lightweight and privacy-preserving.**

***Index Terms*—— Auditing, Cloud Computing, Cloud Storage, Privacy.**

I. INTRODUCTION

Cloud computing is the buzz in the Information Technology world and is presently appears as a hot topic due to its abilities to offer flexible, dynamic IT infrastructures; QoS guaranteed computing environments and configurable software services. National Institute of Standards and technology, agency of US department of Commerce, defines cloud computing as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. The indispensable characteristics of cloud computing includes on-demand self-service, resource pooling, broad network access, rapid elasticity and measured service. Cloud model promotes availability and is composed of three service models and four deployment models. The three service models includes the software as a service (SaaS) which refers to providing on demand applications over the Internet, platform as a service (PaaS) which refers to providing platform layer resources, including operating system support and software development frameworks, and finally the infrastructure as a service (IaaS) which refers to on-demand provisioning of infrastructural resources, usually in terms of VMs. Examples of SaaS providers include Salesforce.com , Rackspace and SAP Business ByDesign. Examples of PaaS providers include Google App Engine, Microsoft Windows Azure and Force.com. IaaS providers include Amazon EC2, GoGrid and Flexiscale [1]. The four deployment models are the Public cloud in which service providers offer their resources as services to the general public, private cloud also known as internal clouds, are designed for exclusive use by a single organization and hybrid cloud is a combination of public and private cloud models that acts on the limitations of each approach.

One of the first cloud offerings was cloud storage and it remains a popular answer. Cloud storage is a model of networked online storage in which the data is stored in virtualized pools of storage that are generally being hosted by the third parties. Cloud storage allows data stored remotely to be temporarily cached on mobile phones, desktop computers, or other Internet-linked devices.

There are several advantages for Cloud computing that it provides several salient features unlike from traditional service computing, which includes multi-tenancy, shared resource pooling, geo-distribution and ubiquitous network access, service oriented, dynamic resource provisioning, self-organizing, utility-based pricing, high scalability , agility, high availability and reliability, reduces the number of hardware components, reduce energy costs for running and cooling hardware components, reduce run time and response time, minimizing the risk of deploying physical infrastructure, lowering the cost of entry, increasing the pace of innovation and more [1]-[3].

Manuscript received March, 2013.
***Neethu Mariam Joseph**, Information Technology, Karunya University., Coimbatore, India.*
***Esther Daniel**, Information Technology, Karunya University., Coimbatore, India,*
***Vasanthi N A**, Department of Computer Science and Engineering, Nehru Institute of Engineering and Technology, Coimbatore, India.*

II. RELATED WORK

There has been an extensive number of works done on untrusted outsourced storage. Employment of cryptographic hash function is one way to enforce the integrity control. Using Merkle tree as the underlying data structure, Yumerefendi and Chase proposed a solution for authenticated network storage. But their handling of updates is computationally expensive. To check the integrity of stored data without download, some researchers have proposed two basic approaches called provable data possession (PDP) (Ateniese et al., 2007) and proofs of retrievability (POR) (Juels, 2007). Ateniese et al. (2007) first proposed the PDP model for ensuring possession of files on untrusted storages and provided an RSA-based scheme for the static case that achieves O(1) communication costs. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the servers for data possession. This property greatly extends application areas of PDP protocol due to the separation of data owners and the authorized users. The "Provable Data Possession" (PDP) model in [9] ensures the possession of data files on untrusted storages. It uses a RSA based homomorphic linear authenticator for auditing outsourced data, but this model leaks the data to external auditors and hence was not provably privacy preserving. Juels et.al in [8] describes a "Proof of Retrievability" (PoR) model, where spot-checking and error correcting codes are used in order to ensure the possession and retrievability. But this approach works only with encrypted data. Improved versions of PoR protocols had been proposed which guarantees private auditability and one which make use of BLS signatures. But these approaches were not privacy-preserving. POR/PDP schemes evolved around an untrusted storage that presents a publicly accessible remote interface to verify the tremendous amount of data. Then comes the TPA based approach to keep online storage honest. This scheme only works for encrypted files which requires the auditor to keep state, and suffers from bounded usage, which potentially brings in online burden to users when the keyed hashes are used up. Schwarz et.al (2010) proposes the use of remote data possession checking to support unlimited times of file integrity verifications on data stored across multiple distributed servers. This approach was based on *erasure-correcting code* and *efficient algebraic signatures*.

Some other works (Li et al., 2006; Ma et al., 2005; Xie et al., 2007; Yavuz and Ning, 2009) considered the problem of auditing the integrity for outsourced data. By openly assuming an order of the records in database, Pang et al. (Ma et al., 2005) used an aggregated signature to sign each record with the information from two neighbouring records in the ordered sequence, which ensures the result of a simple selection query is continuous by checking the aggregated signature. Li et al., 2006; Xie et al., 2007 used a Merkle tree to audit the completeness of query results, but in some extreme cases, the overhead could be as high as processing these queries locally, which can significantly undermine the benefits of database outsourcing. Moreover, to ensure freshness, an extra system is needed to deliver the up-to-date root signature to all clients in a reliable and timely manner [11].

Wang et al. [13] make use of the Merkle Hash Tree to create a public auditing mechanism with fully dynamic data. Zhu et al. [15] used index hash tables to support fully dynamic data. Wang et al. in [14] pondered data privacy with public auditing in the cloud. In their mechanism, the TPA is able to check the integrity of cloud data but cannot gain any private data. Zhu et al. in [11] also aimed at a mechanism to preserve data privacy from the TPA.

III. PROPOSED CONSTRUCTION

This section presents our privacy-preserving interactive audit protocol for outsourced data in cloud storage. Firstly, we will explain the security requirement of cloud audit systems followed by the overview of interactive audit architecture for outsourced data in cloud storage.

A. Security Requirements

The security of the proposed scheme can be stated using a "game" that captures the data possession property [9], [10]. The data possession game between an *adversary A* who might be a malicious CSP and *a verifier V* consists of the following:
Initialization: *Verifier* runs the KeyGen algorithm to generate a key pair (*PK*, *SK*), and sends *PK* to *Adversary*.
Audit*: Adversary* interacts with *Verifier* to get the file and the verification tags set τ. *Verifier* runs the algorithm TagGEN to create the tags set τ, and returns to *Adversary*. Moreover, *Adversary* can request challenge set *{chal}* and return proof *{χ}* to *Verifier*. *Verifier* runs the Verify algorithm and provides the verification results to *Adversary*. The Audit step between *Adversary* and *Verifier* can be repeated polynomial many times.
Challenge. *Adversary* decides on a file *F* previously used during the Audit step, requests a challenge *chal* from *Verifier*, and generates a proof. Upon receiving the proof χ, *Verifier* runs the Verify algorithm and if Verify (*PK*, χ) fails, then *Adversary* has won the game. The *Challenge* step can be repeated polynomial-many times for the purpose of data extraction. The proposed scheme is secure if the probability that any probabilistic polynomial-time (PPT) *adversary A* wins the game is negligible.

B. Design Goals

To allow privacy–preserving public auditing for cloud data storage under this architecture, our protocol design should attain the following security and performance assurances:

- Public Auditability: to allow TPA (or other clients with the help of TPA) to verify the correctness of cloud data on demand without retrieving a copy of whole data or introducing additional on-line burden to the cloud users;
- Verification-correctness: to ensure there exists no cheating CSP that can pass the audit from TPA without indeed storing users' data intact;
- Privacy-preserving: to ensure that there exists no way for TPA to derive users' data from the information collected during the auditing process; and
- Lightweight: to allow TPA to perform auditing with minimum overheads in storage, communication and computation, and to support statistical audit sampling and optimized audit schedule with a long enough period of time.

C. Audit System Architecture for Cloud Storage

Audit system architecture for outsourced data in clouds is shown in figure 1. This architecture can have four entities as follows:

Data Owner (DO): who has large amount of data to be outsourced to cloud

Cloud Service Provider (CSP): who offers data storage service and has enough storage spaces and computation resources

Third Party Auditor (TPA): who has abilities to manage or monitor outsourced data under the entrustment of data owner.

Accorded Applications (AA): who have the right to access and may work on stored data. These applications can be either inside clouds or outside clouds according to the specific necessities.

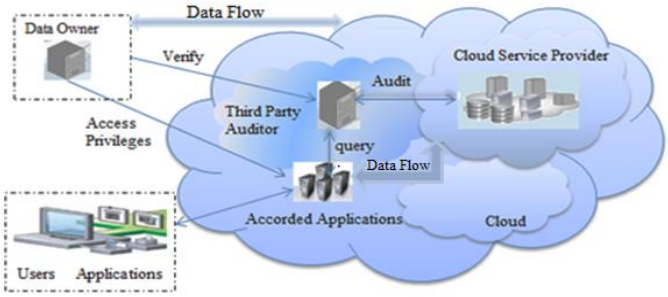

Fig. 1 Audit System Architecture for Cloud Storage

D. Notations And Preliminaries

We would like to introduce the notations and preliminaries.

- F – denotes the data file to be outsourced. It is the sequence of b blocks $m_1, m_2, \ldots m_b \in Z_p$
- H(.), h(.) denotes the cryptographic hash function
- Bilinear Map/Pairing: Let G_1, G_2 and G_T be cyclic groups of prime order p. Let g_1 and g_2 be generators of G1 and G2, respectively. A bilinear pairing is a map $e : G_1 \times G_2 \rightarrow G_T$ with *bilinear*, *non-degenerate*, and *computable* properties.

E. Privacy Preserving Interactive Verification Correctness Scheme

This section presents our proposed privacy preserving verification correctness scheme. The protocol involves the initialization phase, audit phase and verification phase.

The initialization phase uses two algorithms: key generation and tag generation. In the key generation algorithm, each client is assigned a secret key SK, which can be used to generate the tags for the file to be outsourced, and a public key PK, which is used to verify the integrity of those stored files. This public auditing protocol provides the complete outsourcing solution of data stored in cloud, as well provides the data integrity of the data.

In tag generation algorithm, a public verification parameter ω will be generated for each pre-processed file which consists of a set of two values, t and η. The t is the set of tags for each block along with the hash value i.e.; $t = (\delta^{(1)}, t_1, t_2, \ldots t_s)$, where hash value $\delta^{(1)} = H(F_N)$. $\eta = \{\eta_i\}_{i \in [1, b]}$ is the hash index table. The hash index table will take up the design based upon the applications we use. As Zhu describes in [11], for a

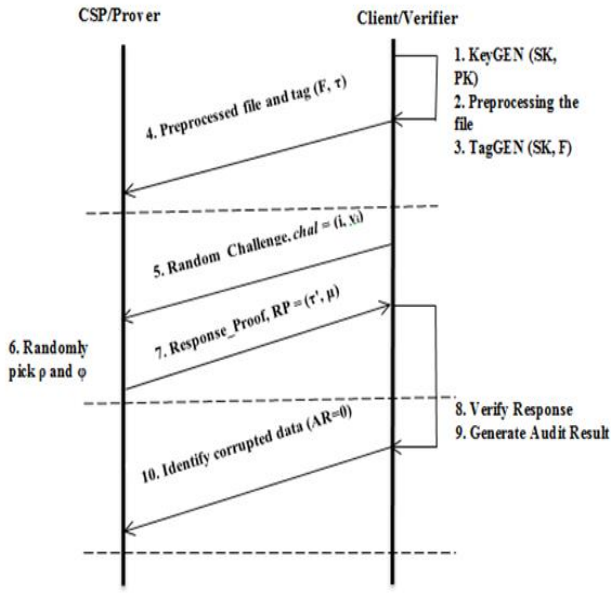

Fig. 2 Interactive Verification Correctness Protocol

Initialization phase:

KeyGEN (1^k): Let Mg = (p, G, G_T, e) be a system of bilinear map group.

i. Randomly select two generators $g_1, g_2 \in G$, where G, G_T are two group of large prime order p.
ii. Chooses a random $sk_1, sk_2 \in Z_p$.
iii. Computes $H_1 = g_2^{sk_1}$ and $H_2 = g_2^{sk_2} \in G$.
iv. Thus, the secret key, SK = { sk_1, sk_2 } and
v. public key is PK = { g_1, g_2, H_1, H_2 }

TagGEN (SK, F):

i. Split the file into b blocks which is again has s sectors.
ii. Choose s random secret for this file $r_1, r_2 \ldots r_s \in Z_p$
iii. Computes $t_i = g_1^{ri} \in G$ for i ∈ [1,s]
iv. Compute hash value $\delta^{(1)} = H_\delta(F_N)$, where $\delta = \sum_{i=1}^{s} ri$ and F_N is the filename.
v. Construct hash index table $\eta = \{\eta_i\}_{i \in [1,b]}$
vi. Calculate its tag as

$$\tau_i \leftarrow (\delta_i^{(2)})^{sk1} \cdot g^{\sum_{j=1}^{s} rj \cdot mi, j \cdot sk_2},$$

where $\delta_i^{(2)} = H_{\delta^{(1)}}(\eta_i)$ and i ∈ [1, b]

vii. Set $t = (\delta^{(1)}, t_1, t_2, \ldots t_s)$
viii. Send outputs $T = (r_1, r_2 \ldots r_s)$, ω = (t, η) to TPA and $\tau = (\tau_1 \ldots \tau_s)$ to CSP

Fig 3 Initialization Phase

static archival file, we can define $\eta_i = S_i$, where S_i is the sequence number of block. For a dynamic file, we can define as $\eta_i = (S_i \| U_i \| R_i)$, where Si is the sequence number of block, Ui is the version number of updates for this block, and to avoid collision we use a random number Ri. In order to ensure the security of stored files the hash index table η is considered very important.

In audit phase, the TPA first retrieves the file tag τ. Now it comes to the "core" part of the auditing process. To generate the challenge message for the audit *chal*, the TPA picks a random *c*-element subset $I = \{C1, C2 \ldots, Cc\}$ of set [1, *b*]. For each element $i \in I$, the TPA also chooses a random value *vi* (of bit length that can be shorter than |*p*|, as in Wang et.al (2011)). The message "*chal*" specifies the positions of the blocks required to be checked. The TPA sends *chal* = $\{(i, \nu_i)\}\ i \in I$ to the server.

Audit Phase: This phase uses the integrity-verification protocol that perform the following operations

Challenge (CSP← TPA): This is the core part of audit algorithm. The *chal* message is used to specify the positions of blocks need to be checked.

i. The verifier chooses a random challenge *chal*, which has c-element subset I= $\{C_1, C_2, \ldots..C_c\}$ of set [1, b].
ii. From this challenge set I, choose z indexes along with z random coefficients $\upsilon_i \in Z_p$
iii. Then *chal* = $\{(i, \upsilon_i)\}_{i \in I}$ be the set of challenge index coefficient pairs
iv. TPA sends this chal to CSP.

Response_Proof (TPA → CSP): CSP performs the following operations

i. CSP chooses a random $\rho \in Z_p$ and
ii. Chooses s random $\varphi_j \in_R Z_p$ for j ε [1,s]
iii. Find $H_1' = H_1^{\rho}$
iv. Find $\psi \leftarrow e\left(\prod_{k=1}^{s} t^{\varphi_j}, H_2\right)$
v. CSP calculates $\tau' \leftarrow \prod_{(i,\nu_i)\in chal} \tau i^{\rho \cdot \nu_i}$
vi. Calculates $\varkappa_j \leftarrow \varphi_j + \rho. \sum_{(i,\nu i)\in chal} \nu_i m_{i,j}$
vii. $\varkappa = \{\varkappa_j\}\ j \in [1,s]$
viii. Prover sends the response $\chi = (H'_1, \psi)$ and $\varsigma = (\tau', \varkappa)$ to verifier

Fig 4: Audit Phase

Upon receiving challenge *chal* = $\{(i, \nu_i)\}\ i \in I$, the server runs Response_Proof to generate a proof of data storage correctness. The linear combination of sampled blocks, *chal* = $\{i, \nu_i\}\ i \in I$. Specifically, the server chooses a random element $\rho \leftarrow Zp$, and calculates $\psi \leftarrow e\left(\prod_{k=1}^{s} t^{\varphi_j}, H_2\right)$. To blind *chal* with ρ, the server computes $\varkappa_j \leftarrow \varphi_j + \rho. \sum_{(i,\nu_i)\in chal} \nu_i m_{i,j}$. Meanwhile, the server also calculates an aggregated authenticator $\tau' \leftarrow \prod_{(i,\nu_i)\in chal} \tau i^{\rho \cdot \nu_i}$. It then sends $\varsigma = \{\tau', \varkappa\}$ as the response proof of storage correctness to the TPA. The figure 4 shows the audit phase.

Audit phase uses the proof generation protocol that has three operations from 5 to 7 as shown in figure 2. This protocol is similar to Schnorr's Σ protocol, which is a zero-knowledge proof system. By using this property, we can ensure that the verification process does not disclose anything. To avoid the leakage of stored data and tags in the verification process, the private data $\{m_{i,j}\}$ are secured by a random $\varphi_j \in \mathbb{Z}p$ and the tags $\{\tau_i\}$ are randomized by a $\rho \in \mathbb{Z}p$. Furthermore, the values $\{\varphi_j\}$ and ρ are protected by the simple methods, i.e., H_1^{ρ} and $t_l^{\varphi j} \in \mathbb{G}$, to prevent the adversaries from gaining those properties. Moreover, it is obvious that this construction admits a short constant-size response $\varsigma = (\tau', \varkappa) \in \mathbb{G} \times Z_p^s$ without being dependent on the size of challenge. That is extremely important for large-size files.

Verification Phase

Verify Response: The verifier can check that the response_proof received are formed correctly by checking the following

$\psi . e(\tau', g_2) \stackrel{?}{=} e\left(\prod_{(i,\nu_i)\in chal} (\delta_j^{(2)})^{\nu_i}, H_1'\right) . e\left(\prod_{j=1}^{s} t_j^{\varphi_j}, H_2\right)$ –(1)

Identify Corrupted Data:

The verifier first validates P using equation (1).

i. The verifier asks the CSP to send $\tau' = \{\tau_i\} 1 \le i \le b$, where $\tau_i = \tau' \leftarrow \prod_{(i,\nu_i)\in chal} \tau_i^{\rho \cdot \nu_i}$, hence the verifier has two lists τList and ϰList
ii. Use a recursive divide and conquer approach, that the verifier can identify the indices of corrupted data.
iii. The τList and ϰList are divided into two halves: τList → (τLList: τRList) and ϰList → (ϰLList: ϰRList).
iv. The verify equation (1) is recursively practiced on τLList with ϰLList and τRList with ϰRList

Thus identifies the corrupted data in the outsourced data

Fig. 5 Verification Phase

The verification phase uses the verification correctness protocol that makes use of verify response operation to check that the response_ proof retrieved is correct or not to generate the audit result, *AR* to be 0 or 1. If the result retrieved to be found 0 then it will check for the corrupted data. It is shown in figure 5, how the corrupted data is identified.

F. Support for Dynamic Operations

As cloud services seem to provide scalability, so we need to support dynamic operations. A dynamic operation includes an insert, update or deletes operations on a single block. While supporting dynamic operations it is essential that the TPA to work with hash-index table η in order to follow the current status of the stored files. This hash-index table has four columns. The first column denotes the actual number i of block b_i, S_i denotes the sequence number of the block, V_i de notes the version number that indicates the number after updating, and finally the R_i that denotes the random number that prevents from collision.

Update(SK, ω, m'_i) : modifies the version number by $V_i \leftarrow \max_{Si = Sj}\{Vj\} + 1$ and chooses a new Ri in $\eta_i \varepsilon \eta$ to get a new η'_i; computes the new hash $\delta_i^{(2)} = H_{\delta}^{(1)}(\text{“}S_i \| V_i \| R_i\text{”})$; by using SK, computes

$$\tau'_i = \left((\delta_j^{(2)})^{sk_1} \cdot \prod_{j=1}^{s} t_j^{m_{i,j}} \right)^{sk_2}$$

where t= {t_j} ε ω, finally outputs O = (η'_i, τ'_i, m'_i).

Delete (SK, ω, m_i): computes the original τ' by m_i and computes the new hash $\delta_i^{(2)} = H_{\delta}^{(1)}(\text{“}S_i \| 0 \| R_i\text{”})$ and $\tau'_i = (\delta_i^{(2)})^{sk1}$; deletes i-th record to get a new ω'; finally outputs O = (η'_i, τ_i, τ'_i)

Insert(SK, ω, m'_i): inserts a new record in *i*-th position of the index-hash table η ε ω and the other records move backward in order; modifies $S_i \leftarrow S_{i-1}$, $V_i \leftarrow \max_{Si = Sj}\{V_j\}$and a random R_i in $\eta_i \varepsilon \eta$ to get a new η'_i ; computes the new hash $\delta_i^{(2)} = H_{\delta}^{(1)}(\text{“}S_i \| V_i \| R_i\text{”})$ and $\tau'_i = (\delta_i^{(2)})^{sk1}$. $\prod_{j=1}^{s} tj^{m'i,j})^{sk2}$, where t= {$t_j$} ε ω, finally outputs O = (η'_i, τ'_i, m'_i).

Verify (U/D/I,O): The application sends the operation type *U*/*D*/*I* and the result *O* is given to *CSP* via a *secure channel.*

- For update or insert operations , CSP must check whether the following is held: for (η'_i, τ'_i, m'_i), e (τ', g_2) ? e($\delta_i^{(2)}$), H_1) .e($\prod_{j=1}^{s} tj^{mi,j}$, H_2); and
- For Delete operation, *CSP* must check whether τ_i is equal to the stored τ_i and e (τ', g_2) ? e ($H_{\delta}^{(1)}(\text{“}S_i \| 0 \| R_i\text{”})$, H_1).

In addition, TPA must replace η_i with the new η'_i and check the completeness $\eta'_i \varepsilon \omega$.

Fig. 6 Dynamic Operations

G. Periodic Sampling Auditing and optimization of parameters

If the audit is performed so frequently it would lead to excess use of network bandwidth and computing resources of TPA, Users, and CSPs. At the same time, too loose audits are not approving to detect deception occurred in outsourced data. Thus, to balance the overhead and reduce the malicious attacks, it is necessary to diffuse the audit tasks during the entire audit cycle. Hence to increase the audit efficiency and to diminish the workload on servers, a sampling-based audit is used. Assume that T is the audit period required to audit the file, which is determined by how important it is for the data owner. The audit period may be usually being assigned as 1 week or 1 month, and for important files may be set as 1 day. Sampling based auditing is used to reduce the server workload and also to increase the efficiency of auditing. Let us assume that *T* is the audit period of the file and *f* is the frequency at which the audit occurs, then the detection probability can be found out as,

$$P_T = 1 - (1- p_b)^{b.w.T.f}$$

We take up that TPA modifies *q* blocks out of b-block file. Then probability of disrupted blocks is $p_b = q/b$. Let the ratio of queried blocks in the total file blocks $w = t/b$ under different detection probability $P = 1 - (1 - p_b)^t$, where t be the number of queried blocks for a challenge in the response_proof.

For a file F with size fs = b. s sectors and probability *p* of sector corruption, the detection probability of verify response is $P \geq 1 - (1- p)^{fs.\ w}$, where $w= t/b$ denotes the sampling probability in verify response.

IV EXPERIMENTAL RESULTS

The file F is used in our experimental results. Storage auditing is a very resource demanding service in terms of computational resource, communication cost and storage space.

Firstly, we compute the performance of our scheme under different parameters, such as file size, sampling ratio, and number sectors per block. The stored files were chosen from 10 KB to 20 MB, the sector numbers were changed from 100 to 500 in terms of the file sizes, and the sampling ratios were also changed from 10% to 50%. The Fig. 7a shows the experimental results which prove that the computation and communication costs rise with increase of file size and sampling ratio.

Then, we relate the performance of initialization phase, for each activity in audit phase and verification phase. We display the experiment results in Fig. 7b, in which the computation and communication costs of the “response_proof” and “verify response” grow with the rise of sampling ratio whereas the core part of protocol “challenge” is slightly changed for sampling ratio.

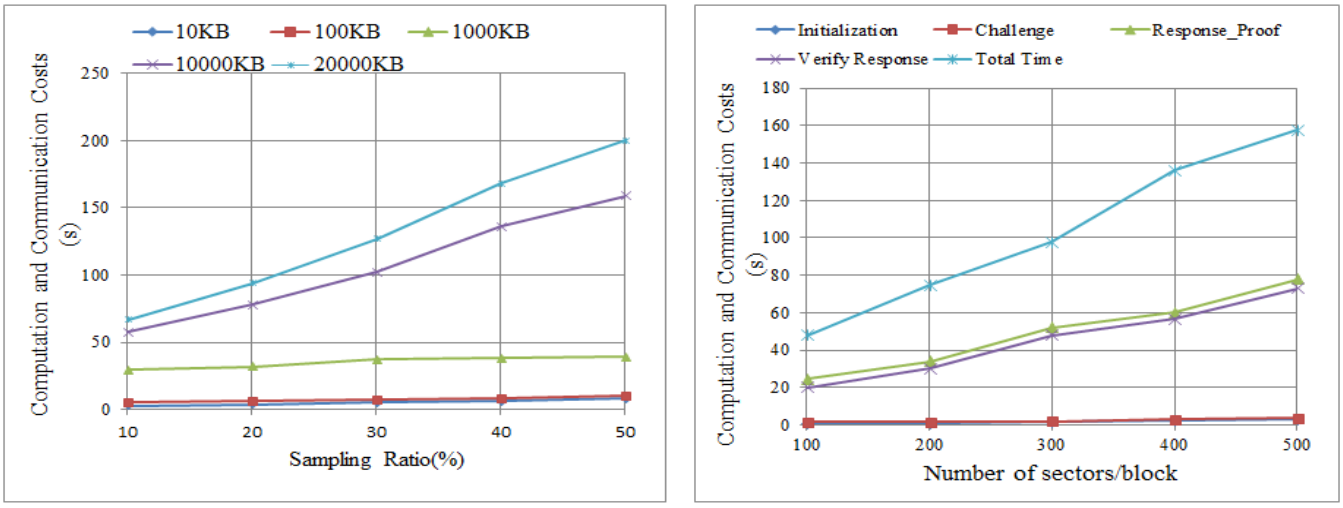

Fig 7 Experimental results under different file size,sampling ratio, and sector number

V CONCLUSIONS

In this paper, we propose a scalable privacy-preserving public auditing system for data storage security in Cloud Computing. We utilize the interactive verification correctness scheme to guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server throughout the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also relieves the users' fear of their outsourced data leakage. Our scheme uses the fragment structure that has block which has sectors where each block corresponds to a tag, so that the storage of signature tags can be reduced by the increase of sectors; a verifier can verify the integrity of file in random sampling approach, which is of utmost importance for large files; these schemes rely on homomorphic properties to aggregate data and tags into a constant-size response, which minimizes the overhead of network communication; and also identifies the corrupted data. Thus our experimental results show that our scheme is provably secure and efficient. This will help the cloud servers which are expected to strongly cope with very large scale data and thus inspire users to adopt cloud storage services more confidently.

REFERENCES

[1] *Qi Zhang, Lu Cheng and Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges," in J Internet Serv Appl, Vol 1, pp. 7–18, April. 2010.*

[2] *Srinivasa Rao V1, Nageswara Rao N. K. and E Kusuma Kumari, "Cloud Computing: An Overview", in Journal of Theoretical and Applied Information Technology, 2005 – 2009.*

[3] *Jayalatchumy D, Ramkumar P, and Kadhirvelu D, "Preserving privacy through data control in a cloud computing architecture using discretion algorithm" in Third International Conference on Emerging Trends in Engineering and Technology, pp.456-461.*

[4] *Rohit Bhadauria. and Sugata Sanyal. 2012. Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques.In International Journal of Computer Applications (June 2012) Volume 47– No.18. 0975 – 888.*

[5] *Traian Andrei, "Cloud Computing Challenges and Related Security Issues", ://www.cse.wustl.edu/~jain/cse571-09/ftp/cloud/index.html. 5/14/2009.*

[6] *Ross A. Lumley, "Cyber Security and Privacy in Cloud Computing: Multidisciplinary Research Problems in Business" in. the George Washington University. Report GW-CSPRI-2010-4, pp. 1-10, December. 2010.*

[7] *Kan Yang and Xiaohua Jia, "Data Storage auditing service in cloud computing: challenges, methods and opportunities," in Springer Science Business Media, Vol 15, pp.409–428, July. 2011.*

[8] *Juels Jr., A. and Kaliski, B.S., "Pors: proofs of retrievability for large files", In: Proceedings of the 2007 ACM Conference on Computer and Communications Security,CCS 2007, pp. 584–597.*

[9] *Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X., "Provable data possession at untrusted stores", In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 598–609.*

[10] *H. Shacham and B. Waters, "Compact proofs of retrievability," Cryptology ePrint Archive, Report 2008/073, 2008.*

[11] *Yan Zhu. Hongxin Hu. Gail-Joon Ahn. and Stephen S. Yau, "Efficient audit service outsourcing for data integrity in clouds". In The Journal of Systems and Software 85, pp.1083– 1095, December. 2011.*

[12] *Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847–859, 2011.*

[13] Wang, Q., Wang, C., Li, J., Ren, K., Lou, W, " Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computin", In: Proc. European Symposium on Research in Computer Security. pp. 355–370. Springer-Verlag (2009).

[14] *Wang, C., Wang, Q., Ren, K., Lou, W, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In: Proc. IEEE INFOCOM. pp. 525– 533 (2010)*

[15] *Zhu, Y.,Wang, H., Hu, Z., Ahn, G.J., Hu, H., Yau, S.S., "Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds", In: Proc. ACM Symposium On Applied Computing. pp. 1550–1557 (2011)*

Neethu Mariam Joseph has received the B.Tech degree in Information Technology from Anna University of Technology, Coimbatore. She is currently pursuing her M.Tech degree in the Network and Internet Engineering at Karunya University, Coimbatore. Her research interests include Computer Networks, Network Security and Cloud Computing.

Esther Daniel is working as Assistant .Professor in the Department of Information Technology in Karunya University. She has done her Bachelor of Engineering from Bharathiar University and Master of Engineering from Karunya University. Her teaching experience spans to 7years. Her research area includes Network Security, Cloud Computing etc. She has 6 publications at national and International level. She is doing part time research in Information Technology at Anna University of Technology, Coimbatore.

Dr. N.A.Vasanthi is currently the Professor and Head of Department of Computer Science at Nehru Institute of Engineering and Technology. She received the B.E degree in ICE and M.E. degree in CSE, both from Bharathiar University, Coimbatore, India and Ph.D from Anna University. She is a life member of CSI and ISTE. She has over two decades of teaching and research experience. Her area of interest is Wireless Networks, Pervasive Computing and Soft Computing. She has published papers in International and National Journals.