Indian Journal of Science and Technology, *Vol 9(18), DOI: 10.17485/ijst/2016/v9i18/90458, May 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

An Efficient Scheme to Establish Content and Audit Liveness in the Cloud

T. Santhoshi Rupa* and K. V. V. Satyanarayana

Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation (KLU), Vaddeswarm, Guntur - 520002, Andhra Pradesh, India; tallapragadasanthu@kluniversity.in, kopparti@kluniversity.in

Abstract

Background/Objectives: Data Storage in the cloud based services are evident everywhere in this day and age starting from Google doc based file management services, file storage services etc. Various schemes with respect to protecting this data range from ciphering services, auditing services, migration services etc. All these services help a cloud user to preserve their data based on a certain time threshold. **Methods/Statistical Analysis:** The current context of the paper concentrates on delivering an efficient data audit service. Previously these Audit services used to reside within the cloud domain but due to untrustworthiness of many cloud operators, Outside Auditor (OA) implementations were gearing up to validate cloud stored data. **Findings:** Earlier these OA's used to download users data at their end to check their integrity leading to users privacy problems. Also it happens to be a time complex operation. Hence we propose an efficient file chunk sampling to detect anomalous changes in the cloud stored data. Index Hash structures play a vital role in this aspect. **Applications/Improvements:** This process is an iterative procedure instead of one time operation happening at regular intervals and importantly preserving user's privacy.

Keywords: Audit, Cloud Repository, Distributed Hash Tables, Index Hash Table, Local & Remote Copy, Public Storages

1. Introduction

Remote Storage environment offered by Cloud is both flexible and convenient in terms of scalability with respect to users and their data. Many of these cloud services happens to be on demand, low cost approach, location independent storage etc. Many reputed cloud service providers manage and maintain the necessary infrastructure for these storage operations. Users' data is vulnerable to failures or manipulations by the cloud itself. The following illustrations highlight our concern.

A user X stores the data in the cloud which comprises of five files. As long as there is an activity on the files, the cloud hosts the data. The activity is defined in terms of access, download, partial download etc. When such activity lapses the cloud issues a warning to the user to maintain activity. The user has many solutions. Some of them are being providing activity to their content by self, migrating their data to a more reliable storage provider etc. If these initiatives from the user are absent, the cloud owner determines the survivability of users' data. Most probably they will take a backup, snapshot of the content and reclaim the users' storage space for their purposes. These operations most of the times manipulate users data. In another scenario, insiders within the cloud manipulate user data without their permission. These manipulations can be defined as editing, renaming, replacing etc. Data auditing in cloud empowers both cloud owners and users with so many utilities such as activity traces, unauthorized data access, application logs, virtual machine breaches etc. Data access tracking is vital for either cloud owners or data owners equally and these proposed privacy preserving auditing services protects the cloud owners from data sharing's for validations and at the same time provides enough information to the users with respect to the status (Alive, Alive but manipulated, Deleted etc.) of their data. As of now many countries don't have regulations with these auditing services but some developed countries do have Authorized Regulations which these audit services and cloud owners should comply with.

*Author for correspondence

The advantage of the proposed system is not just the privacy concerns of the user but prior solutions proves to be complex and impractical because for checking the integrity of a file to determine whether it is compromised or intact requires to download the whole data from cloud owner which increases communication cost exponentially. This will be more when the size of the file is in Giga bytes which it usually does. Our proposed audit is efficient and the cloud owner should uphold the following parameters.

- Providing bridge between cloud and Auditor: This module allows an OA to estimate the correctness of a client's data in the cloud and report that estimation back to the client.
- Dynamic measures: Our proposed three-stage protocol ensures there are no security breaches with respect to user data stored in clouds.
- Periodic check: Detecting data manipulations or deletions due to anomalous behaviors (Inside or outside) from cloud perspective should not be a onetime affair but it should be an iterative procedure.
- Effective exploration: This allows the OA to implement auditing under cloud users supervision of their data. It should support the exploration results as evidences to determine a security compromise. Technically in this module it requires to use a hash table structure.
- Minimal foot print: The auditing process performed by OA will be having a minimal computational foot print in terms of lower processing cost, reduced storage overhead.

The work done so far with respect to checking file intactness of a cloud user's data is based on file hashing schemes[1]. Most of these schemes require the physical presence of the file to generate hash. This compromises user's data security and can be termed as security breach. The communication cost increases exponentially when we consider huge file sizes. But without having a local copy, outside auditors cannot work and hence these schemes can be scrapped. So alternatives with respect to OA's functionality with reduced communication costs were required to provide assurance to cloud users.

Other schemes implementing proof of retrievability concepts[2] provided the file intactness measures of cloud user's data but they happen to be one time schemes and are not feasible in a real time scenario. We need an evolved iterative measure especially when the cloud is an untrusted party[3]. Many of these schemes and our scheme do require from the cloud side to have a bridge API that can provide an interface to check file intactness of cloud user's data.

Some other schemes implemented a chunk sampling mechanisms[4] at OA domain. To illustrate better this scheme considers an input file which is directly partitioned into X parts and then each part generates a unique signature tag. To sustain data protection[5], the size of the part must be same as the size of the security cipher that is considered which should be around 20 Bytes. This means that a 2 MB file will be partitioned into 1 Lakh parts and simultaneously will lead to 1 Lakh unique signature tags and the resultant size of these storage tags will amount to approximately 2 MB which is a huge storage overhead. So building a file intactness checking system using this scheme is inefficient and impractical and thus can be overridden.

The theory of Encryption based on attributes also called Attribute Based Encryption[6] was used in order to secure information with respect to cloud computing domain. In this access policy users tend to define the attributes that are required to construct a public key that will be shared among users which can be used to encrypt or decrypt contents. The attributes range from Data Owners identity[7] (involving credentials) and file Meta data that includes name, extension, size, creation date etc. prior approaches implemented a centralized approach which happens to be a monotonic structure to generate and handle key distributions.

Cryptographic protocols named ring, mesh and group signatures are available to utilize. Ring signature is not an appropriate option as there would be several users in cloud environments. Group signature option has an already existing group and hence not possible in cloud environment. Mesh signature technique do not ensure if the message is from single or from multiple users collaborating together.

2. Proposed Work

We propose an iterative three step cloud data auditing architecture that can be performed on a remote cloud user's data. It comprises of four main entities which we define using the following terminologies:

- A Cloud User (CU) who has a large volume of data stored in a cloud.
- A Cloud Owner (CO) who provides a platform for all Cloud Users (CU) to store their data.
- An Outside Auditor (OA) who is equipped with a remote file monitoring capabilities and is permitted

and invited by a cloud user to perform the audit tasks on their cloud account. The OA's architecture should be in such a way that they can implement file intactness checks on remote cloud users data irrespective of the cloud platform being a SAAS (Software As A Service), PAAS (Platform As A Service), IAAS (Infrastructure As A Service). We assume the OA is autonomous and his file intactness check reports are reliable even though there is a chance that an OA can be a malicious reporting entity. We leave this malicious OA resolution as future work. The OA is also equipped with an iterative, periodic file intactness checking mechanisms[12] which aids a cloud user for immediate and subsequent (Quite possible) checkpoints.

Each step of the three stage protocol involved in our demonstrative prototype is reflected in the following Figure 1, Figure 2 and Figure 3.

The status of the OA cannot be established using any existing means so we propose the following extension.

Algorithm: Bitset implementation of the live-in check
Bool is_live_in(var a, int q) {
intdef = get_def_block_num (a);
intmax_dom = get_max_num (def);
if (q <= def || max_dom< q)
return false;
int t= bitset_next_set (T[q], def + 1);
while (t <= max_dom) {
for (each u in def_use chain of a)
if (bitset_is_set(R[t],u))
return true;
t = get_max_num (t) + 1;

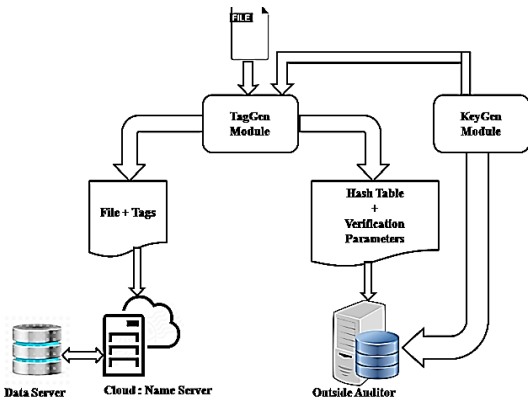

Figure 1. Tag Generation for Cloud Out-Sourced storage File.

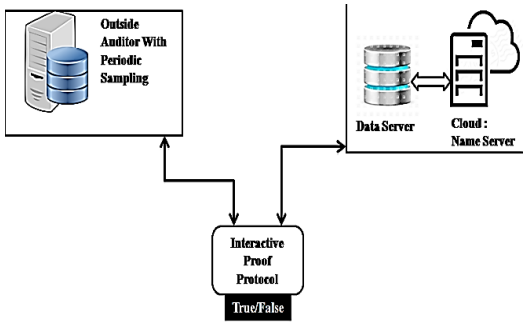

Figure 2. Periodic Sampling Audit.

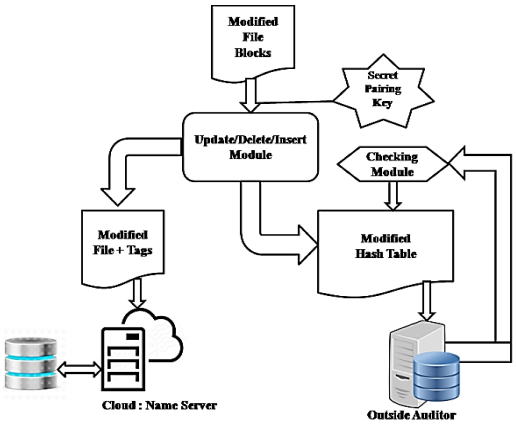

Figure 3. Dynamic Data Operations and Audit.

t = bitset_next_set(T[q], t);
}
Return false;
}

The OA's internal workings is comprised of the following 3 stages

2.1 File Signature Generation

Any File having some internal content will exhibit a unique signature for its overall content or different chunks of signatures for each equal file partition. We use the former file signature representation using a renowned (SHA 256 bit) hash function. For every registered cloud user a unique token for the user will be generated by the cloud server which can be used as an authentication to perform the signature generation process as they act as validation parameters in generating a hash structure.

2.2 Periodic File Chunk Audit

This process is quite challenging for the outside auditor as it involves processing user's dynamic session values to

obtain a gateway to fetch Meta data information of uses data.Since this is an iterative procedure, a handshake agreement between the cloud owners or the cloud server and the OA will ease this task.

2.3 Cloud user-File Signature Hash Structure

We incorporate a two dimensional Hash structure that contains all cloud users in one dimension and all their stored file hashes in another dimension for quick processing.

For supporting the interactivity between OA and Co which happens to be dynamic real time events, the two dimensional hash structure is used which contains stored file hashes information that is generated to trace the file block modifications between the current version of the file and with a previous version of the file which is stored at auditors file repository which is designated in Figure 3 Architecture. Normally the hash structure contains block number, chunk size, chunk hash value. The obvious sub stages involved in the 3 stage process are mentioned here:

- **KeyGen**

This module involves generating a unique chunk signature present in the file to be validated.

- **Meta Tag Generation**

This module involves pairing the key gen value with data owner's token representation for session handling and to support periodic file intactness checks.

- **Provenance**

This module involves estimating prior file intactness checks with new dynamically obtained file intactness check and to decide on the issue of file compromise resulting in a security breach. It contains four other stages which are mentioned in the following list

- Commitment
- Challenge
- Response
- Check

3. Performance Analysis

We design the implementation on a standard PC with the configurations Core 2 Duo, 4 GB RAMS with windows operating system. The cloud storage server deployed in this host supports parallel uploads from the user. In our

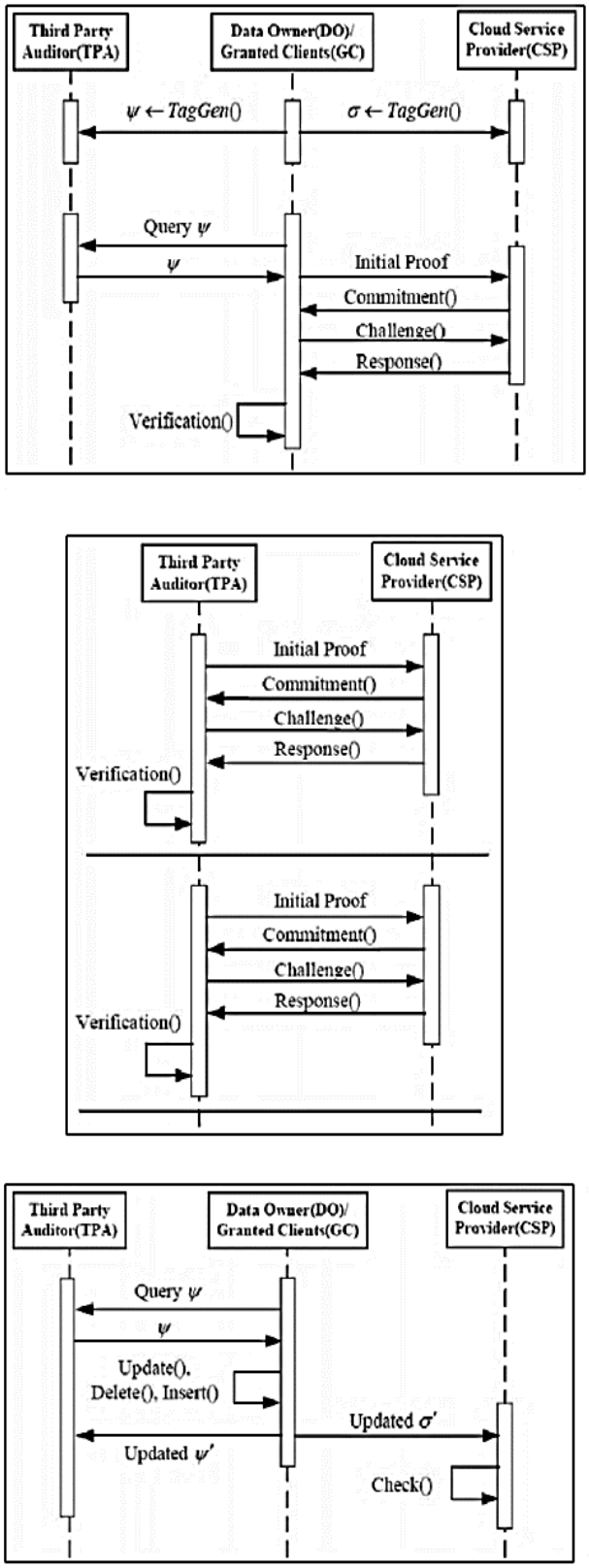

Figure 4. The above three-stage representation suggest the flow of different formats of data between auditing entities.

implementation, we uploaded a series of files ranging from 5 Kb to 1 MB to perform audits with the help of the OA. The results were considered with and without the availability of OA.

The latency factor during the estimation of audits in the absence of OA is found to be higher and the resultant notification encountered by the user happens to be machine exceptions which are not feasible to an end user. So we proposed the above witness check that outputs a Boolean flag that determines the availability of OA.

So equipped with these audit challenges and liveness checks, the CO can offer services with respect to endurance factor

So a similar configuration replicated on a virtual box Ubuntu image instance replicates the same results and we are confident an Amazon EC2 VM[9] instance will offer the same.

4. Conclusion

In this paper many issues were discussed for file intactness check of cloud user's data which is a result of untreatable activities of cloud owners. So we proposed an iterative three stage protocol to ensure file correctness simultaneously respecting user's privacy with respect to their data at an outside auditors end. It also possesses a uniquely distinguishable periodic validation process which is flexible with respect to a user's perspective compared to a one time affair. Also we extend this proposal with an outside auditor liveness check procedure to determine the availability of an auditor at all stages because this extension aids a user in deciding whether to continue to the same auditor or change to a new one. The problem and handling of a malicious OA's presence within the architecture is left for future work.

5. References

1. Hsiao HC, Wang KH, Kikuchi H, Perrig A, Sun HM, Yang BY. A study of user-friendly hash comparison schemes. Proceedings of Annual Computer Security Applications Conference (ACSAC); 2009. p. 105–14.
2. Juels A. PORs: Proofs of retrievability for large files. Proceedings of ACM Conference Computer and Communications Security (CCS'07); 2007. p. 584–97.
3. Megiba Jasmine R, Nishibha GM. Public cloud secure group sharing and accessing in cloud computing. Indian Journal of Science and Technology. 2015 Jul; 8(15). Doi no:10.17485/ijst/2015/v8i15/75177
4. Ateniese G, Curtmola R, Herring J, Kissner L, Peterson ZNJ, Song DX. Provable data possession at untrusted stores. Proceedings of 14th ACM Conference Computer and Communication Security. 2007. p. 598–609.
5. Manjusha R, Ramachandran R. Secure authentication and access system for cloud computing auditing services using associated digital certificate. Indian Journal of Science and Technology. 2015 Apr; 8(S7). Doi no:10.17485/ijst/2015/v8iS7/71223.
6. SenthilKumari P, Nadira Banu Kamal AR. Optimal Integrity policy for encrypted data in secure storage using cloud computing. Indian Journal of Science and Technology. 2016 Mar; 9(8). Doi no:10.17485/ijst/2016/v9i8/87923
7. Shacham H. Compact proofs of retrievability. Proceedings of 14th International Conference Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'08); 2008. p. 90–107.
8. Erway C, Papamanthou C, Tamassia R. Dynamic provable data possession. Proceedings of 16th ACM Conference on Computer and Communication Security.
9. Amazon Web Services. Amazon s3 Availability Event: 2008 Jul 20. Available from: http://status.aws.amazon.com/s3-20080720.html
10. Mowbray M. The fog over the Grimpen Mire: Cloud computing and the law. Technical Report. HPL-2009-99.
11. Yavuz A. BAF: An efficient publicly verifiable secure audit logging scheme for distributed systems. Proceedings of Annual Computer Security Applications Conference (ACSAC); 2009. p. 219–28.
12. Ateniese G, Mancini LV, Sudik GT. Scalable and efficient provable data possession. Proceedings of 4th International Conference on Security and Privacy in Communication Networks (Secure Comm); 2008. p. 1–10.
13. B. V. S. Srikanth, V. Krishna Reddy. Efficiency of Stream Processing Engines for Processing BIGDATA Streams. Indian Journal of Science and Technology. 2016; 9(14).