IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 5 (Jul. - Aug. 2013), PP 30-33
www.iosrjournals.org

Privacy and Integrity Preserving in Cloud Storage Devices

V. Tejaswini[1], S.k. Prashanth[2], Dr. N.sambasiva Rao[3], C. Satya kumar[4],
[1]*M.Tech (C.S.E), VCE, A.P, India.*
[2]*Associate professor, VCE, A.P, India.*
[3]*Professor, VCE, A.P, India.*
[4]*Associate professor, VCE, A.P, India.*

Abstract: *Cloud computing is an internet based model which allows the user all the "services on demand". The services provided by cloud are infrastructure, software and platform. Cloud Computing has great potential of providing robust computational power to the society at reduced cost. Despite the tremendous benefits, security is the primary obstacle that prevents the wide adoption of this promising computing model, especially for customers when their confidential data are consumed and produced during the computation. Wang proposed enabling public audit ability and data dynamics for storage security in cloud computing which provides data integrity and dynamic operation but cannot assure confidentiality. Here in this scheme we are providing a new method which provide confidentiality and also manage the storage space in cloud server. In this we are providing confidentiality by using cryptographic algorithm and storage space is managed by using complete binary tree by arranging all the files by their size and checking the integrity using TPA(third party auditor). Thus providing integrity, memory management as well as confidentiality in cloud computing.*
Keywords *– cloud computing, data integrity, data storage, public audit ability*

I. INTRODUCTION

Cloud Computing provides convenient on-demand network access to a shared pool of configurable computing resources that can be deployed with great efficiency and minimal management overhead. One primary advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constraint devices. By outsourcing the workloads into the cloud, customers could enjoy the unlimited computing resources in a pay-per-use manner without committing large capital outlays in the purchase of both hardware and software. In spite of the incredible benefits, outsourcing computation to the commercial public cloud is also miserly customers' direct control over the systems that consume and produce their data during the computation, which certainly brings in new security challenges towards this promising computing model [2]. Numerous trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the "software as a service" (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high-quality services from data and software that reside solely on remote data centers. Although envisioned as a promising service platform for the Internet, this new data storage paradigm in "Cloud" brings about many challenging design issues which have profound influence on the security and performance of the Overall system. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers.

On the one hand, the outsourced computation workloads often contain sensitive information, such as the business financial records, proprietary research data, etc. To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing so as to provide end-to- end data confidentiality assurance in the cloud. However, ordinary data encryption techniques in essence prevent cloud from performing any meaningful operation of the underlying plaintext data, making the computation over encrypted data a very hard problem. On the other hand, the operational details inside the cloud are not transparent enough to customers [4]. As a result, there exist various motivations for cloud server to behave unfaithfully and return incorrect results. For example, for the computations that require a large amount of computing resources, there are huge financial incentives for the cloud to be "lazy" if the customers cannot tell the correctness of the output.

Besides, possible software bugs, hardware failures, or even outsider attacks might also affect the quality of the computed results. Thus, we argue that the cloud is basically *not secure* from the point of customers. Without providing a mechanism for secure computation outsourcing, i.e., to protect the sensitive input and output information and to validate the integrity of the computation result, it would be hard to expect cloud customers to turn over control of their workloads from local machines to cloud solely based on its economic savings as well as resource flexibility. For practical consideration, such a design should further ensure that customers perform fewer amounts of operations than completing the computations by themselves directly.

Recent researches made steady advances in "privacy and integrity preserving in cloud computing".

Cloud Computing treating the cloud as an intrinsically insecure computing platform from the viewpoint of the cloud customers, we must design mechanisms that not only protect sensitive information by enabling computations with encrypted data, but also protect customers from malicious behaviors by enabling the validation of the computation result. Such a mechanism of general secure computation outsourcing was recently shown to be feasible in theory, but to design mechanisms that are practically efficient remains a very challenging problem.

Consider the large size of the outsourced data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files. In order to solve the problem of data integrity, many schemes are proposed under different models [2], [3], [4], [5], [6], [7], [8]. In all these works, great efforts are made to design solutions that meet various requirements of each and every user such as: high scheme efficiency, stateless verification and retrievability of data, etc.

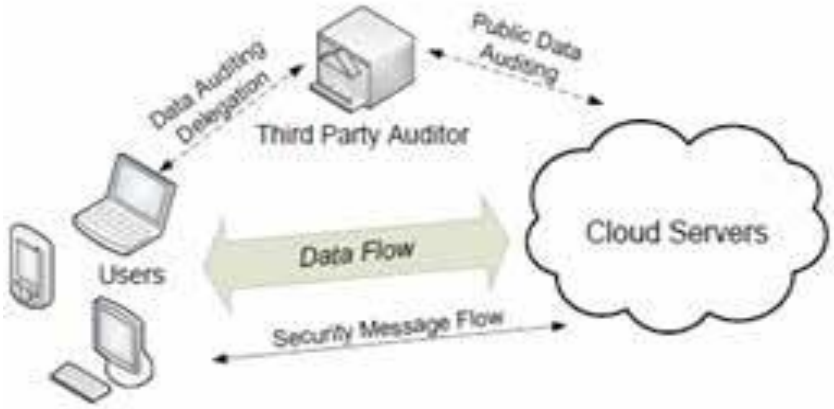

Fig 1. Cloud computing using TPA

II. RELATED WORK

Recently, much of growing interest has been pursued in the context of remotely stored data verification [1], [3], [4], [5], [6], [7], [8]. ateniese et al. [2] are the first to consider public auditability in their defined "provable data possession" model for ensuring possession of files on untrusted storages. In their scheme, they utilize RSA-based homomorphic tags for auditing outsourced data, thus public auditability is achieved.

However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems.. Ateniese et al.

Similar to, they only consider partial support for dynamic data operation. Juels and Kaliski [3] describe a "proof of retrievability" model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on archive service systems. Specifically, some special blocks called "sentinels" are randomly fixed into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [12], the number of queries a client can perform is also a fixed priori, and the introduction of precomputed "sentinels" prevents the development of realizing dynamic data updates. Erway et al. were the first to explore constructions for dynamic provable data possession. They extend the PDP model in [2] to support provable updates to stored data files using rank-based authenticated skip lists. This scheme is essentially a fully dynamic version of the PDP solution.

Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both confidentiality and memory management is not achieved. How to achieve a secure and efficient design to integrate these two important mechanisms for data storage service remains an open exigent task in Cloud Computing.

III. Threat Model

The threats can come from two different sources: internal attacks and external attacks.

For internal attacks, a CSP can be self-interested, untrusted and possibly mal-icious. Not only it desire to move data that is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide the data loss incidents due to management errors, failures etc.For external attacks, data integrity threats may comes from outsiders who are beyond the CSP.

IV. Design Goals

1. Storage integrity: to ensure users that their data are stored appropriately and kept intact all the time in the cloud for the multiple files.
2. Confidentiality: providing security for the data by using encryption techniques.
3. Memory space management in cloud server using complete binary tree.

V. Proposed Work Construction

Before storing the data in the server the client first calculate the hash values for each and every file and keeps with him. After calculating the hash values with polynomial hashing technique, the client places his file in the server. To check the integrity we are constructing a complete binary tree where all the ''F_n'' files are arrange as shown in the figure based their file size. If we want to find integrity for '$f_1,f_2,f_3,\ldots..f_{n-1}$'' any of the files then the TPA downloads the total number of files (i.e the files which client challenges the server to find the integrity) and performs the integrity checking simultaneously by creating threads for each file. Thus hash values which the client get by the server is compared with the hash values which are with the client. If both the hash values are matched then their exist integrity else the file is corrupted.

f1 to fm
.
.
.

fm to fn

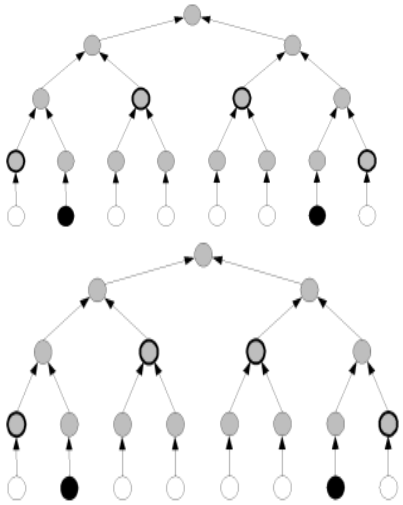

Fig 2. Complete Binary Tree

Algorithm 1.

Step 1 : Create an N object for 'N' files
Step 2 : Create a thread for each file
Thread t1=new Thread(f1);
Thread t2=new thread (f2);
--

Thread tn=new Thread(fn);
Step 3 : t1.start(f1);
t2.start(f2);

---
tn.start(fn);

Algorithm 2

Step 1 : Create a Class using Runable Interface
Step 2: Implement a run() method
-Process take place (Calculates hash values)
-Sends hash value to client.

For N types of files we create N objects and for each object we create a thread. If client wants to check an integrity for F2, F4 then the TPA downloads the total file and creates a thread T2, T4 respectively and simultaneously checks the integrity using runnable interface. For each thread we calculate hash values and send it to the server. The server sends the hash values to the client and the client compares both the hash values and checks the integrity. If the hash values of the server matches with the client then it said to be integrity exist.

VI. CONCLUSION

This approach is very secure and attains the integrity without the users burden. All the integrity checking is performed by the Third party auditor(TPA) so that the clients involvement is reduced. Here the memory is also managed as all the files are arranged to a complete binary tree. In order to attain the confidentiality we are using some cryptographic algorithm and sending the files to the server. So we are achieving integrity, confidentiality and supports memory management.

References

[1] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing, *European Symp. Research in Computer Security (ESORICS '09),* 2009, pp. 355-370.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, Provable Data Possession at Untrusted Stores, *ACM Conf. Computer and Comm. Security (CCS '07),* 2007, pp. 598-609.

[3] A. Juels and B.S. Kaliski Jr., Pors: Proofs of Retrievability for Large Files, *ACM Conf. Computer and Comm. Security (CCS '07),* 2007, pp. 584-597.

[4] H. Shacham and B. Waters, Compact Proofs of Retrievability, *Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08),* 2008,pp. 90-107.

[5] K.D. Bowers, A. Juels, and A. Oprea, Proofs of Retrievability: Theory and Implementation,Report 2008/175, Cryptology ePrint Archive, 2008.

[6] M. Naor and G.N. Rothblum, The Complexity of Online Memory Checking, *Ann. IEEE Symp. Foundations of Computer Science (FOCS '05),* 2005, pp. 573-584.

[7] E.-C. Chang and J. Xu, Remote Integrity Check with Dishonest Storage Server, *European Symp. Research in Computer Security (ESORICS '08),* 2008, pp. 223-237.

[8] M.A. Shah, R. Swaminathan, and M. Baker, Privacy-Preserving Audit and Extraction of Digital Contents,*Report 2008/186, Cryptology ePrint Archive*, 2008.