www.ioirp.com
International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 1, Issue – 9. August 2015

UNRESTRICTED AUDITABILITY AND STATISTICS CHANGING ASPECTS FOR REFUGE IN CLOUD STORAGE

Miss. N.Valarmathi/Assistant Professor
Department of Computer Science and Engineering
Cheran College of Engineering
K.Paramathi,Karur-639 111.TamilNadu,India
Email.Id: mathi.natarajan@gmail.com

***Abstract*---Cloud computing refers to the provision of computational resources on demand via a computer network and also it has been envisioned as the next generation architecture of IT enterprise. The principle behind the cloud is that any computer connected to the internet is connected to the same pool of computing power, applications, and files. The application software and data's are resides in the centralized large data center's but the center or server is not fully trustworthy. To ensure the integrity of data storage in cloud computing third party auditing (TPA) is used on behalf of the client to avoid the frequent checking by client to verify the data's are stored in correct location, and its indeed intact. Moreover, to achieve economies of scale, which is one of the advantages of cloud computing. The support for data dynamics via the most general forms of data operation such as block modification, insertion, and deletion, is also a significant step toward practicality, since services in cloud computing are not limited to archive or backup data. This paper aims at modifying the existing proof of storage models by manipulating the Logarithmic merkle hash tree construction for block tag authentication, to achieve efficient data dynamics and BLS algorithm for error correction, bilinear aggregate signature to support multiuser auditing. This paper achieve highly efficient and provably secure. Cloud environment is created using eucalyptus open source.**

***Keywords*— Data storage, public auditability, data dynamics, cloud computing**

I. INTRODUCTION

Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). A cloud service has three distinct characteristics that differentiate it from traditional hosting. It is sold on demand, typically by the minute or the hour; it is elastic - a user can have as much or as little of a service as they want at any given time and the service is fully managed by the cloud service provider. The advantage of cloud is cost savings. The prime disadvantage is security.

Cloud computing is used by many software industries now a days. Sometimes the cloud service provider may hide the data corruptions to maintain the reputation. To avoid this problem, we introduce an effective third party auditor to audit the user's outsourced data when needed. The security is achieved by signing the data blocks. Signing is performed using BLS algorithm. We utilized public key based homomorphic authenticator with random masking to achieve privacy preserving auditing protocol. PA performs the auditing task for each user i.e. single auditing. This increases the auditing time and computation overhead. The technique of Bilinear Aggregate Signature is used to achieve batch auditing i.e. multiple auditing tasks simultaneously. Earlier works performs auditing only for static data. We enhance the system with dynamic operations on data blocks i.e.) data update, append and delete. Reed Solomon technique is a best error correction technique which ensures the correctness of the data stored in cloud.

Cloud storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators, in the background,

virtualize the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers.Cloud storage services may be accessed through a web service application programming interface (API), or through a Web-based user interface.

A.RELATED WORKS

Ateniese et al., [2007]They proposed provable data possession model for ensuring possession of files on untrusted storages. They utilize RSA-based homomorphic tags for auditing outsourced data, thus public auditing is achieved. It 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. Subsequent work a dynamic version of the prior PDP scheme. However, the scheme imposes a prior bound on the number of queries and does not support fully dynamic data operation. It allows only very basic block operations with limited functionality, and block insertions cannot be supported.

Wang et al., [2009] this paper considers dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. They only consider partial support for dynamic data operation. It use publicly verifiable homomorphic authenticators built from BLS signatures on which the proofs can be aggregated into small authenticator value, and public retrievabiltiy is achieved. This scheme provides integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. To achieve a secure and efficient design to seamlessly integrate data storage services.

A.Jules and B.S. Kaliski [2007]This paper presents spot-checking and error-correcting codes are used to ensure both possession and retrievabiltiy of data files on archive service systems. Some special sentinels are randomly embedded into data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. The number of queries a client can perform is also a fixed priori, and the introduction of precompiled sentinels prevents the development of realizing dynamic data updates. In addition, public auditability is not supported in their scheme. The client can periodically challenge the storage server to ensure the correctness of the cloud data, and the original files can be recovered by interacting with the server.

Erway et al., [2009] they extended the PDP model to support provable updates to stored data files using rank- based authenticated skip lists. This scheme is essentially for block insertion; theyeliminate the index information in the tag computation and employ skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. The existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud.

AmanBakshi et al.,[2010] This paper presented a futuristic platform to provide dynamic resource pools, virtualization, and high availability and enables the sharing, selection and aggregation of geographically distributed heterogeneous resources for solving large-scale problems in science and engineering. But with this ever developing cloud concept, problems are arising from this golden solution in the enterprise arena. Preventing intruders from attacking the cloud infrastructure is the only realistic thing the staff, management and planners can foresee. Regardless of company size and volume and magnitude of the cloud, they explained how manoeuvre IT virtualization strategy could be used in responding to a denial of service attack.

OrganizationThe rest of the paper is organized as follows. In section III, we define the system model,our design goals. Then, we present the proposed scheme in section IV Finally, we conclude in section V.

II. SYSTEM DESIGN

Cloud computing components are classified as 1) Cloud User (CU), 2) Cloud Service provider (CSP) and 3) Cloud server,(CS)

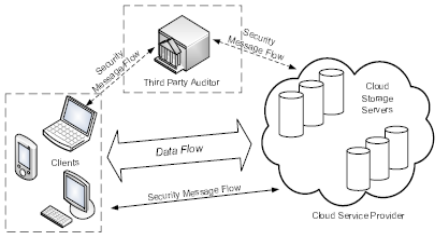


Fig. 1: Cloud data storage architecture

Client: an entity, which has large data files to bestored in the cloud and relies on the cloud fordata maintenance and computation, can be eitherindividual consumers or organizations.

Cloud Storage Server (CSS): an entity, which is managedby Cloud Service Provider (CSP), has significantstorage space and computation resource tomaintain the clients' data.

Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is *trusted* to assess and expose risk of cloud storage Services on behalf of the clients upon request.

In the cloud paradigm, by putting the large data fileson the remote servers, the clients can be relieved ofthe burden of storage and computation. As clients nolonger possess their data locally, it is of critical importancefor the clients to ensure that their data are being correctly stored and maintained. That is, clients shouldbe equipped with certain security means so that they can periodically verify the correctness of the remote dataeven without the existence of local copies. In case thatclients do not necessarily have the time, feasibility orresources to monitor their data, they can delegate themonitoring task to a trusted TPA.

The third party auditor (TPA), who has expertise and capabilitiesthat cloud users do not have and is trusted to assess the cloudstorage service security on behalf of the user upon request. Usersrely on the CS for cloud data storage and maintenance. They mayalso dynamically interact with the CS to access and update theirstored data for various application purposes. The users may resortto TPA for ensuring the storage security of their outsourced data,while hoping to keep their data private from TPA. We considerthe existence of a semi-trusted CS as does. Namely, in most oftime it behaves properly and does not deviate from the prescribed
protocol execution. However, during providing the cloud datastorage based services, for their own benefits the CS might neglect to keep or deliberately delete rarely accessed data files which
belong to ordinary cloud users. Moreover, the CS may decide tohide the data corruptions caused by server hacks or Byzantinefailures to maintain reputation. We assume the TPA, who is inthe business of auditing, is reliable and independent,and thus hasno incentive to collude with either the CS or the users during theauditing process. TPA should be able to efficiently audit the clouddata storage without local copy of data and without bringing inadditional on-line burden to cloud users.

III. DESIGN GOALS

TPA must be designed for Integrity checking without local copy maintaining in client. It should work in distributed environment. It Must achieve all types of security requirements in cloud storage. Support data dynamic operation. Highly efficient and resilient to attackers.

To create an efficient BLS algorithm for signing the data. To achieve the data dynamics for block data logarithmic Merkle hash tree is used. To perform multiple auditing tasks bilinear aggregate signature is used.

IV. PROPOSED SYSTEM

To effectively support public auditability without having to retrieve the data blocks themselves, Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. In this scheme PKC based homomorphic authenticator to equip the verification protocol with public auditability. Assume that file F (potentially encoded using Reed-Solomon codes) is divided into n blocks $m_1, m_2, \ldots, m_n$, where $m_i \in Z_p$ and p is a large prime.

Let $e: G \times G \rightarrow G_T$ be a bilinear map, with a hash function $H: \{0,1\}^* \rightarrow G$, viewed as a random oracle .Let g be the generator of G. h is a cryptographic hash function.
The BLS signature scheme consists of three phases,

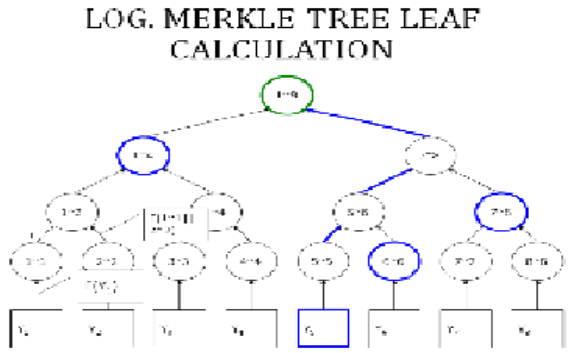
1. In the key generation phase, a sender chooses a random integer x € Z and computes $y = g_1{}^x \in G_1$. The private key is x and the public key is y.

2. Given a message $m \in \{0,1\}$ in the signing phase, the sender first computes $h = h(m) \in G_1$, where h () is a hash function, and then computes $\sigma = h^x \in G1$. The signature of m is σ.

3. In the verification phase, the receiver first computes $h = h(m) \in G_1$ and then check whether. If the verification succeeds, then the message m is authentic.

A. *Logarithmic merkle hash tree*

This algorithm is implemented in TPA. Merkle trees have found many uses in theoretical cryptographic constructions, having been specifically designed so that a leaf value can be verified with respect to a publicly known root value and the authentication data of the leaf. This authentication data consists of one node value at each height, where these nodes are the siblings of the nodes on the path connecting the leaf to the root. The Merkle tree traversal problem is the task of finding an efficient algorithm to output this authentication data for successive leaves. The trivial solution of storing every node value in memory requires too much space. On the other hand, the approach of computing the authentication nodes on the round they are required will be very expensive for some nodes. The challenge is to conserve both space and computation by amortizing the cost of computing such expensive nodes.



1. Set $leaf = 0$.
2. **Output:**
   - Compute and output $\Phi(leaf)$ with $LEAFCALC(leaf)$.
   - For each $h \in [0, H-1]$ output $\{Auth_h\}$.
3. **Refresh Auth Nodes:**
   For all h such that 2^h divides $leaf + 1$:
   - Set $Auth_h$ be the sole node value in $Stack_h$.
   - Set startnode $= (leaf + 1 + 2^h) \oplus 2^h$.
   - $Stack_k.initialize(startnode, h)$.
4. **Build Stacks:**
   Repeat the following $2H - 1$ times:
   - Let l_{min} be the minimum of $\{Stack_h.low\}$.
   - Let $focus$ be the least h so $Stack_h.low = l_{min}$.
   - $Stack_{focus}.update(1)$.
5. **Loop:**
   - Set $leaf = leaf + 1$.
   - If $leaf < 2^H$ go to Step 2

F
ig 4.1.1 Algorithm for Log.Merkle hash tree

The upcoming needed authentication nodes are computed as in the classic traversal, but the various stacks do not all receive equal attention. Each TREEHASH instance can becharacterized as being either not started, partially completed, or completed. Our schedule to prefers to complete Stack for the lowest h values first, unless another stack has a lower tail node. We express this preference by defining l min be the minimum of the h values {Stack.low}, then choosing to focus our attention on the smallest level h attaining this minimum. (settingStack.low = 1 for completed stacks effectively skips them over). In other words, all stacks must be completed to a stage where there are no tail nodes at height h or less before we start a new Stack TREEHASH computation.

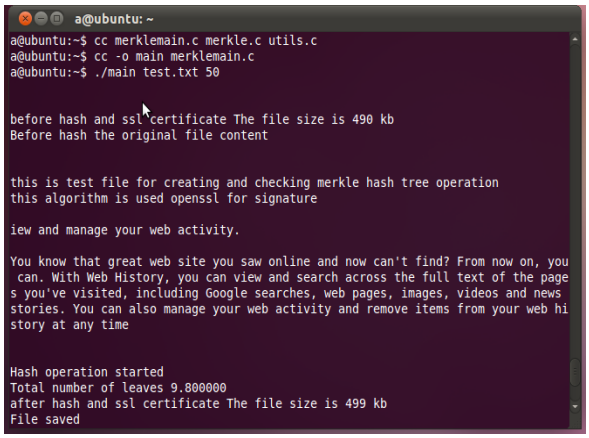


Fig 4.1.2integrity checking of log merkle algorithm at sending.

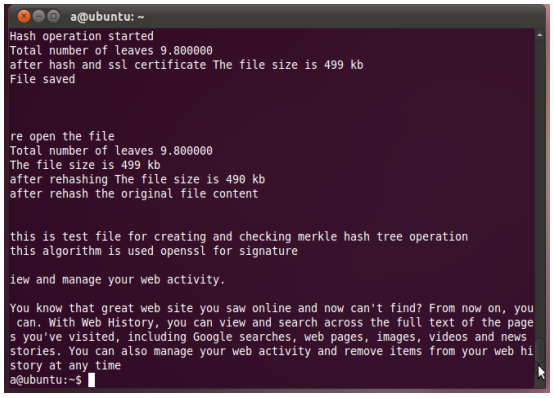


Fig 4.1.3 integrity check at receiving

B.Multiple Batch Auditing

TPA may concurrently handle multiple auditing delegations upondifferent users' requests. The individual auditing of these tasksfor TPA can be tedious and very inefficient. Given K auditingdelegations on K distinct data files from K different users, it ismore advantageous for TPA to batch these multiple tasks together and audit at one time. Keeping this natural demand in mind, the proposed technique of bilinear aggregate signature, whichsupports the aggregation of multiple signatures by distinct signerson distinct messages into a single signature and thus provideseefficient verification for the authenticity of all messages. Usingthis signature aggregation technique and bilinear property, it aggregate K verification equations into a single one, so thatthe simultaneous auditing of multiple tasks can be achieved.

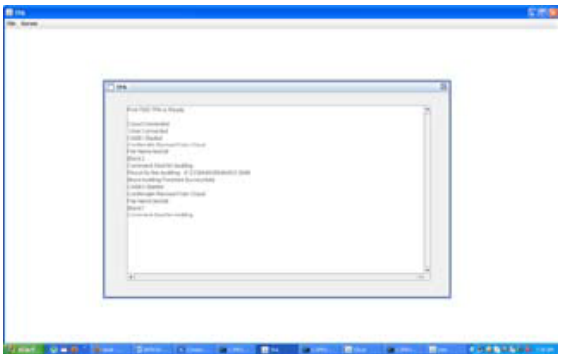


Fig 4.2.1.multiple auditing

C. Designs for Distributed Data Storage Security.

To further enhance the availability of the data storage security, individual user's data can be redundantly stored in multiple physical locations. That is, besides being exploited at individual servers, data redundancy can also be employed across multiple servers to tolerate faults or server crashes as user's data grows in size and importance. It is well known that erasure-correcting code can be used to tolerate multiple failures in distributed storage systems. In cloud data storage, we can rely on this technique to disperse the data file F redundantly across a set of n = m+kdistributed servers. A [m+k, k]-Reed-Solomon code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m+kdata and parity vectors. By placing each of the m + k vectors on a different server, the original data file can survive the failure of any k of the m+kservers without any data loss. Such a distributed cryptographic system allows a set of servers to prove to a client that a stored file is intact and retrievable.

V. CONCLUSION

This system is the first to support scalable and efficient public auditing in the Cloud storage. The technique of Bilinear Aggregate signature is used to achieve batch auditing and also multi-client auditing, where TPA can perform multiple auditing tasks simultaneously. The data in the cloud does not remain static. In this system make use of Logarithmic merkle hash tree algorithm for supporting data dynamic operations. Its performance is high when compared to other. auditing services. Log. Merkle hash tree algorithm t increase the speed of TPA to provide high performance. It supports secure and efficient dynamic operations on data blocks stored in the cloud, including: data update, delete and append.

References

1. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of CCS'07*. New York, NY, USA: ACM, 2007, pp. 598–609.

2. A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in *Proc. of CCS'07*. New York, NY, USA: ACM, 2007, pp. 584–597.

3. H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. of ASIACRYPT'08*. Melbourne, Australia: Springer-Verlag, 2008, pp. 90–107.

4. K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," Cryptology ePrint Archive, Report 2008/175, 2008.

5. M. Naor and G. N. Rothblum, "The complexity of online memory checking," in *Proc. of FOCS'05*, Pittsburgh, PA, USA, 2005, pp. 573–584.

6. E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in *Proc. of ESORICS'08*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 223–237.

7. M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

8. A. Oprea, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in *Proc. of NDSS'05*, San Diego, CA, USA, 2005.

9. T. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *Proc. of ICDCS'06*, Lisboa, Portugal, 2006, pp. 12–12.

10. Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," in *Proc.OfIEEE INFOCOM'09*, Rio de Janeiro, Brazil, April 2009, pp. 954–962.