Research Paper

Volume : 1 | Issue : 5 | Oct 2012 • ISSN No 2277 - 8179

# Enabling Dynamic Data In Cloud Storage

Computer Science

**KEYWORDS :** Data Storage, Data Dynamics, Cloud Computing

**A.Mounika** M. Tech, Dept. Of Computer Science and Engineering, Kakatiya Institute Of Technology and Sciences, Warangal AP, India

**C.Srinivas** Associate Professor, Dept. Of Computer Science and Engineering, Kakatiya Institute Of Technology and Sciences, Warangal AP, India

## ABSTRACT

*Cloud computing is used in next generation of IT Enterprise. In Cloud Computing software applications and databases are moved to centralized large data centers, where the management of the data and services will not be fully trust, this brings out many new security challenges. This work studies the problem of assuring the integrity of data storage in cloud computing. In this we consider a third party auditor (TPA) instead of cloud client to verify the integrity of the dynamic data stored in the cloud. The dynamic data is supported via many forms of data operation such as block modification, insertion and deletion. The previous works often lacks the support of dynamic data operations. In this , to achieve efficient dynamic data , we improve the existing proof of storage models by manipulating the classic merkle hash tree construction for block tag authentication.*

## I. INTRODUCTION

Several trends are opening in the era of cloud computing, which is an internet-based development and use of computer technology. The 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 large scale. At same time, the increasing network bandwidth and reliable, flexible network connections make it even possible that clients can now subscribe high-quality services from data and software that reside by not involving anything else on remote data centers. One of the biggest worry with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. Consider the large size of the worked out electronic 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 checking, many schemes are proposed under different systems and security models. In all these works, great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc. Considering the role of the verifier in the model, all the schemes presented before fall into two categories: private auditability and public auditability. Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources. In the cloud, the clients themselves are unreliable or may not be able to afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public auditability, which is expected to play a more important role in achieving economies of scale for Cloud Computing. Moreover, for efficiency consideration, the outsourced data themselves should not be required by the verifier for the verification purpose.

**· Previous Work**

The major concern among previous designs is that of supporting dynamic data operation for cloud data storage applications. In Cloud Computing, the remotely stored electronic data might not only be accessed but also updated by the clients, e.g., through block modification, deletion, insertion, etc. Unfortunately, the state of the art in the context of remote data storage mainly focus on static data files and the importance of this dynamic data updates has received limited attention so far. Moreover, the direct extension of the current provable data possession (PDP) or proof of retrievability (PoR) schemes to support data dynamics may lead to security loopholes. Although there are many difficulties faced by researchers, it is well believed that supporting dynamic data operation can be of vital importance to the practical application of storage outsourcing services.

## II. PROPOSED SYSTEM

In view of the key role of public auditability and data dynamics for cloud data storage, we propose an efficient construction for the seamless integration of these two components in the protocol design.

**Our contribution can be summarized as follows:**

1. We motivate the public auditing system of data storage security in Cloud Computing, and propose a protocol supporting for fully dynamic data operations, especially to support block insertion, which is missing in most existing schemes.
2. We extend our scheme to support scalable and efficient public auditing in Cloud Computing. In particular, our scheme achieves batch auditing where multiple delegated auditing tasks from different can be performed simultaneously by the TPA.
3. We prove the security of our proposed construction and justify the performance of ourscheme through concrete implementation and comparisons with the state of the art.

Assume the outsourced data file F consists of a finite ordered set of blocks m1,m2, . . .,mn. One straightforward way to ensure the data integrity is to precompute MACs for the entire data file. Specifically, before data outsourcing, the data owner precomputes MACs of F with a set of secret keys and stores them locally. During the auditing process, the data owner each time reveals a secret key to the cloud server and asks for a fresh keyed MAC for verification. This approach provides deterministic data integrity assurance

Merkle hash tree authentication of data elements. We treat the leaf nodes h(x1),...,h(xn) as the left-to-right sequence. Straight forwardly as the verification covers all the data blocks. However, the number of verifications allowed to be performed in this solution is limited by the number of secret keys. Once the keys are exhausted, the data owner has to retrieve the entire file of F from the server in order to compute new MACs, which is usually impractical due to the huge communication overhead. Moreover, public auditability is not supported as the private keys are required for verification. Another basic solution is to use signatures instead of MACs to obtain public auditability. The data owner precomputes the signature of each block an sends both F and the signatures to the cloud server for storage. To verify the correctness of F, the data owner can adopt a spot-checking approach, i.e., requesting a number of randomly selected blocks and their corresponding signatures to be returned. This basic solution can provide probabilistic assurance of the data correctness and support public auditability. However, it also severely suffers from the fact that a considerable number of original data blocks should be retrieved to ensure a reasonable detection probability, which again could result in a large communication overhead and greatly affects system efficiency. Notice that the above solutions can only support the case of static data, and none of them can deal with dynamic data updates. Recently, much of growing interest has been pursued in the context of remotely stored data verification [2], [3], [4], [5], [6], [7], [8], [9], [10] 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. In their subsequent work [12], Ateniese et al. propose a dynamic version of the prior PDP scheme.

However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported. In [13], Wang et al. consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [12], 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 embedded 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. In addition, public auditability is not supported in their scheme. Shacham and Waters [4] design an improved PoR scheme with full proofs of security in the security model defined in [3]. They use publicly verifiable homomorphic authenticators built from BLS signatures [16], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static data files. Erway et al. [14] 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. To support updates, especially for block insertion, they eliminate the index information in the "tag" computation in Ateniese's PDP model [2] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear. Although 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 Computing. This project provides authentication on the data transaction and also to show the authorized data receiving also. To those reasons, we are developing this project with the following modules:

### A. Authentication
In this module, Admin searches the client system which is connected in the network and then gets the IP Address of the particular client system. Admin makes connection between client and his system after the client gives permission to the admin to connect with itself. In Authentication module new user going to register their details and registered user login their account. So unauthorized user can't access this service. In this module, the server enters the username and password in the text fields in the login form, and then authorized person gets the permission to access the Cloud Server and TPA application.

### B. Key provider
In this module we only secure our transmission of the data because the key provider will check the every single packet, whether it has the key specification or not. If any other intruders shall have to receive any data from us this application not accept to receive because of the key specification.

### c. Block Tag creation
In this module we select one folder ,then we split the selected folder to 'N' no of tags. For the secure purpose we are splitting files into tags. After splitting the file tags we give tag id for individual tag.

### D. File Transfer
In this module, we send the splitted tag file to the receiver, along with tag id list. In this case send the file in highly securely. So no one can understand the original file information. In general cloud service having security, for that reason doing all process in the above format. So user send request, tag List, and modification for specific process to Third party Auditor.

### E. Cloud Storage Server
Inside of cloud many resources are there, for identifying particular server, send all those information ,based on these information cloud identify one server and doing the Auditing process. After auditing process information will stored in cloud. Here user check their details within cloud by the help of TPA. So TPA is intermediate between user and server (Cloud).

### F. TPA verification
In this, TPA have request, Tag list, username, based on these information TPA will going to verify the user data content in storage server. Compare with TPA file and cloud stored file if it's any mismatched file acquiring TPA send that information to the user.

## III. SIMULATION RESULTS

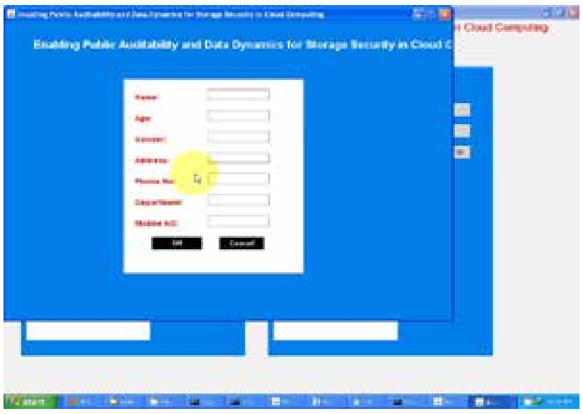
Enabling Public Auditability and Data Dynamics for Storage Security in Cloud C
OK
Cancel

Fig.1 User Registration

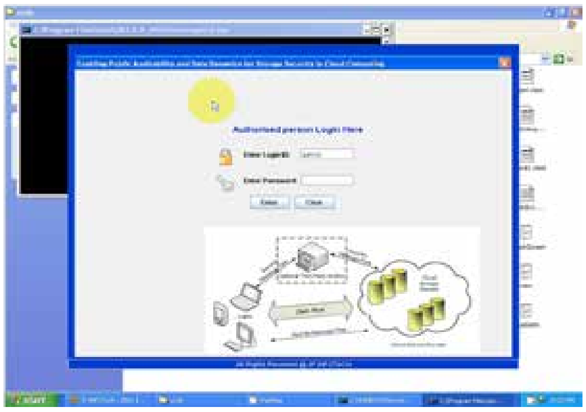

Fig.2 User Login

Volume : 1 | Issue : 5 | Oct 2012 • ISSN No 2277 - 8179

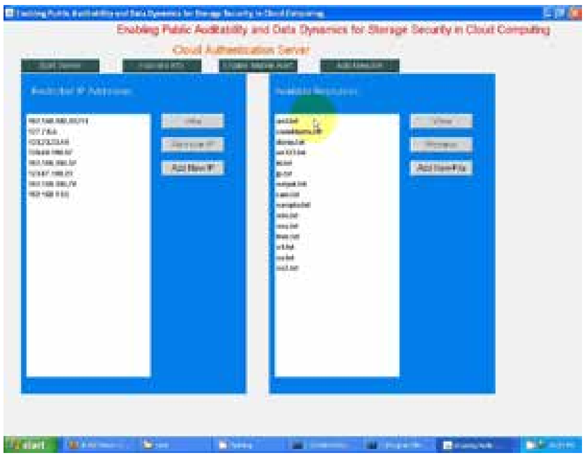
Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing
Cloud Authentication Server

Fig. 3 Block Insertion Of IP Addresses And Files

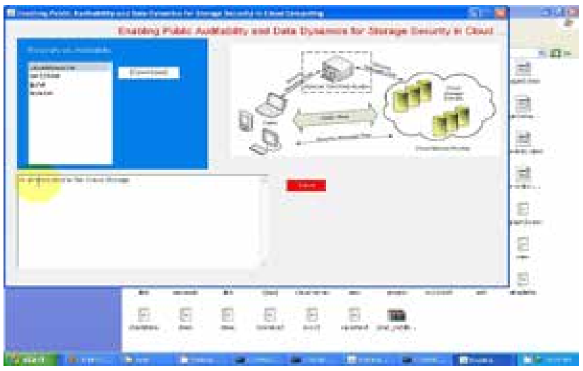
Enabling Public Auditability and Data Dynamics for Storage Security in Cloud

Fig . 4 Block Modification Of Files

## IV.CONCLUSION

To ensure cloud data storage security, it is critical to enable a TPA to evaluate the service quality from an objective and independent perspective. Public auditability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications. Another major concern is how to construct very-fication protocols that can accommodate dynamic data files. In this paper, we explored the problem of providing simultaneous public auditability and data dynamics for remote data integrity check in Cloud Computing. Our construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multiuser setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

## REFERENCE

[1] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing,” Proc. 14th European Symp. Research in ComputerSecurity (ESORICS ’09), pp. 355-370, 2009. | [2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” Proc. 14th ACM Conf. Computer and Comm. Security (CCS’07), pp. 598-609, 2007. | [3] A. Juels and B.S. Kaliski Jr., “Pors: Proofs of Retrievability for Large Files,” Proc. 14th ACM Conf. Computer and Comm. Security | (CCS ’07), pp. 584-597, 2007. | [4] H. Shacham and B. Waters, “Compact Proofs of Retrievability,” Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT ’08), pp. 90-107, 2008. | [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,” Proc. 46th Ann. IEEE Symp. Foundations of Computer Science (FOCS ’05), pp. 573-584, 2005. | [7] E.-C. Chang and J. Xu, “Remote Integrity Check with Dishonest Storage Server,” Proc. 13th European Symp. Research in Computer Security (ESORICS ’08), pp. 223-237, 2008. | [8] M.A. Shah, R. Swaminathan, and M. Baker, “Privacy-Preserving Audit and Extraction of Digital Contents,” Report 2008/186, Cryptology ePrint Archive, 2008. | [9] A. Oprea, M.K. Reiter, and K. Yang, “Space-Efficient Block Storage Integrity,” Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS ’05), 2005. | [10] T. Schwarz and E.L. Miller, “Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage,” Proc. 26th IEEE Int’l Conf. Distributed Computing Systems (ICDCS ’06), p. 12, 2006. | [11] Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” Proc. IEEE INFOCOM, pp. 954-962, Apr. 2009. | [12] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. Fourth Int’l Conf. Security and Privacy in Comm. Networks (SecureComm ’08), pp. 1-10,2008. | [13] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring Data Storage Security in Cloud Computing,” Proc. 17th Int’l Workshop Quality of Service (IWQoS ’09), 2009. | [14] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, “Dynamic Provable Data Possession,” Proc. 16th ACM Conf. Computer and Comm. Security (CCS ’09), 2009. | [15] K.D. Bowers, A. Juels, and A. Oprea, “Hail: A High-Availability and Integrity Layer for Cloud Storage,” Proc. 16th ACM Conf. Computer and Comm. Security (CCS ’09), pp. 187-198, 2009. | [16] D. Boneh, B. Lynn, and H. Shacham, “Short Signatures from the Weil Pairing,” Proc. Seventh Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT ’01), pp. 514-532, 2001.