Efficient Data Retrieval in Cloud Storage

S. Jayanthi
Assistant Professor,
Dept of CSE, University College of Engineering
Trichirappalli, India

K. Devika
M.E Student,
Dept of CSE, University College of Engineering
Trichirappalli, India

Abstract:- **Cloud computing is a model for enabling well-situated, on- demand network access to a collective pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned out with minimal management effort or service provider interaction. Maintaining integrity of outsourced data is a trivial task in cloud storage mechanism. This paper proposes a public auditing scheme for validating the outsourced data. A Proxy is introduced to reduce the online burden of the data owners. The Regenerating code mechanism used in this paper provides fault tolerance by striping data across multiple servers, while using less repair traffic than traditional erasure codes during failure recovery. Thus the end user can download data stored in cloud without worry about the correctness of data.**

General Terms:- Random Key Generation, Advanced Encryption Standard Algorithm, Code Regeneration Mechanism, Storage Efficient Retrieval

Keywords:- Cloud Storage, Third Party Audit, Code regeneration, Proxy, Data Retrieval

1. INTRODUCTION

Cloud computing is defined as a type of computing that relies on-*sharing computing resources* rather than having local servers or personal devices to handle applications. Cloud computing is comparable to grid computing, a type of computing where unused processing cycles of all computers in a network are harnesses to solve problems too intensive for any stand-alone machine. The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive online computer games. To do this, cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing. Cloud storage is now gaining popularity because it offers a flexible on-demand data outsourcing service with appealing benefits: relief of the burden for storage management, universal data access with location independence, and avoidance of capital expenditure on hardware, software, and personal maintenance etc., Nevertheless, this new paradigm of data hosting service also brings new security threats toward user data, thus making individuals or enterprisers still feel hesitant. It is noted that data owners lose ultimate control over the fate of their outsourced data; thus, the correctness, availability and integrity of the data are being put at risk. On the one hand, the cloud service is usually faced with a broad range of internal/external adversaries, who would maliciously delete or corrupt users' data; on the other hand, the cloud service providers may act dishonestly, attempting to hide data loss or corruption and claiming that the files are still correctly stored in the cloud for reputation or monetary reasons. Thus it makes great sense for users to implement an efficient protocol to perform periodical verifications of their outsourced data to ensure that the cloud indeed maintains their data correctly. Many mechanisms dealing with the integrity of outsourced data without a local copy have been proposed under different system and security models up to now. The most significant work among these studies are the PDP (provable data possession) model and POR (proof of retrievability) model, which were originally proposed for the single-server scenario by Ateniese et al. [1] and Juels and kaliski [2] respectively. Considering that files are usually striped and redundantly stored across multi-servers or multi-clouds, [3]-[9] explore integrity verification schemes suitable for such multi-servers or multi clouds setting with different redundancy schemes, such as replication, erasure codes, and, more recently, regenerating codes. In this paper, we focus on the integrity verification problem in regenerating-code-based cloud storage, especially with the functional repair strategy [10].Similar studies have been performed by Bo Chen et al. [6] and H. Chen el al. [7] separately and independently. [6] extended the single-server CPOR scheme(private version in [11]) to the regenerating code - scenario; [7] designed and implemented a data integrity protection (DIP) scheme for FMSR [12] -based cloud storage and the scheme is adapted to the thin-cloud setting. However, both of them are designed for private audit, only the data owner is allowed to verify the integrity and repair the faulty servers. Considering the large size of the outsourced data and the user's constrained resource capability, the tasks of auditing and reparation in the cloud can be formidable and expensive for the users. The overhead of using cloud storage should be minimized as much as possible such that a user does not need to perform too many operations to their outsourced data (in additional to retrieving it). In particular, users may not want to go through the complexity in verifying and reparation. The auditing schemes imply the problem that users need to always stay online, which may impede its adoption in practice, especially for long-term archival storage. To fully ensure the data integrity and save the users' computation resources as well as online burden, we propose a public auditing scheme for the regenerating-code-based cloud storage, in which the integrity checking and regeneration are implemented by a third party auditor and a semi-trusted proxy separately on behalf of the data owner. Instead of directly adapting the existing public auditing scheme to the multi-server setting, we design a novel authenticator, which is more appropriate for regenerating codes. Besides, we "encrypt" the coefficients to protect data privacy against the auditor, which is more lightweight than applying the proof blind technique in and data blind method in. Several challenges and threats spontaneously arise in our new system model with a proxy and security analysis shows that our scheme works well with these problems. Specifically, our contribution can be summarized by the following aspects: We design a novel homomorphic authenticator based on BLS signature which can be generated by a couple of secret keys and verified publicly. Utilizing the linear subspace of the regenerating codes, the authenticators can be computed efficiently. Besides, it can be adapted for data owners equipped with low end computation devices (e.g. Tablet PC etc.) in which they only need to sign the native blocks. To the best of our knowledge, our scheme is the first to allow privacy-preserving public auditing for regenerating code- based cloud storage. The coefficients are masked by a PRF (Pseudorandom Function) during the Setup phase to avoid leakage of the original data. This method is lightweight and does not introduce any computational overhead to the cloud servers or TPA. Our scheme completely releases data owners from online burden for the regeneration of blocks and authenticators at faulty servers and it provides the privilege to a proxy for the reparation. Optimization measures are taken to improve the flexibility and efficiency of our auditing scheme; thus, the storage overhead of servers, the computational overhead of the data owner and communication overhead during the audit phase can be effectively reduced. Our scheme is provable secure under random oracle model against adversaries. Moreover, we make a comparison with the state of the art and experimentally evaluate the performance of our scheme. The easiest way to do this is simply to download the template, and replace the content with your own material.

2. RELATED WORK

The problem of remote data checking for integrity was first introduced in Ateniese et al. [1]and Juels and Kaliski [2] gave rise to the similar notions provable data possession (PDP) [1] and proof of retrievability (POR) [2], respectively. Ateniese et al. proposed a formal definition of the PDP model for ensuring possession of files on untrusted storage, introduced the concept of RSA-based homomorphic tags and suggested randomly sampling a few blocks of the file. In their subsequent work, they proposed a dynamic version of the prior PDP scheme based on MAC, which allows very basic block operations with limited functionality but block insertions. Simultaneously, Erway et al. [13] gave a formal framework for dynamic PDP and provided the first fully dynamic solution to support provable updates to stored data using rank-based authenticated skit lists and RSA trees. To improve the efficiency of dynamic PDP, Wang et al. [14] proposed a new method which uses merkle hash tree to support fully dynamic data. Many storage systems rely on replication to increase the availability and durability of data on untrusted storage systems. At present, such storage systems provide no strong evidence that multiple copies of the data are actually stored. Storage servers can collude to make it look like they are storing many copies of the data, whereas in reality they only store a single copy. We address this shortcoming through multiple-replica provable data possession (MR-PDP) [3]: A provably-secure scheme that allows a client that stores t replicas of a file in a storage system to verify through a challenge-response protocol that (1) each unique replica can be produced at the time of the challenge and that (2) the storage system uses t times the storage required to store a single replica. MR-PDP extends previous work on data possession proofs for a single copy of a file in a client/server storage system. Using MR-PDP to store t replicas is computationally much more efficient than using a single-replica PDP scheme to store t separate, unrelated files (e.g., by encrypting each file separately prior to storing it). Another advantage of MR-PDP is that it can generate further replicas on demand, at little expense, when some of the existing replicas fail. HAIL (High-Availability and Integrity Layer) [4], a distributed cryptographic system that permits a set of servers to prove to a client that a stored file is intact and retrievable. HAIL strengthens, formally unifies, and streamlines distinct approaches from the cryptographic and distributed-systems communities. Proofs in HAIL are efficiently computable by servers and highly compact— typically tens or hundreds of bytes, irrespective of file size. HAIL cryptographically verifies and reactively reallocates file shares. It is robust against an active, mobile adversary, i.e., one that may progressively corrupt the full set of servers. We propose a strong, formal adversarial model for HAIL, and rigorous analysis and parameter choices. We show how HAIL improves on the security and efficiency of existing tools, like Proofs of Retrievability (PORs) deployed on individual servers. We also report on a prototype implementation. Remote Data Checking (RDC) [5] is a technique by which clients can establish that data outsourced at untrusted servers remains intact over time. RDC is useful as a prevention tool, allowing clients to periodically check if data has been damaged, and as a repair tool whenever damage has been detected. Initially proposed in the context of a single server, RDC was later extended to verify data integrity in distributed storage systems that rely on replication and on erasure coding to store data redundantly at multiple servers. Recently, a technique was proposed to add redundancy based on network coding, which offers interesting tradeoffs because of its remarkably low communication overhead to repair corrupt servers. Unlike previous work on RDC which focused on minimizing the costs of the prevention phase, we take a holistic look and initiate the investigation of RDC schemes for distributed systems that rely on network coding to minimize the combined costs of both the prevention and repair phases. We propose RDC-NC, a novel secure and efficient RDC scheme for network coding-based distributed storage systems. RDC-NC mitigates new attacks that stem from the underlying principle of network coding. The scheme is able to preserve in an adversarial setting the minimal communication overhead of the repair component achieved by network coding in a benign setting. We implement our scheme and experimentally show that it is computationally inexpensive for both clients and servers.

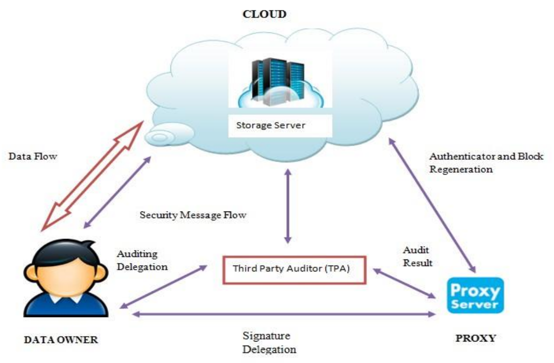
3. SYSTEM MODEL

We consider the auditing system model for Regenerating-Code-based cloud storage, which involves four entities: the data owner, who owns large amounts of data files to be stored in the cloud; the cloud, which are managed by the cloud service provider, provide storage service and have significant computational resources; the third party auditor (TPA), who has expertise and capabilities to conduct public audits on the coded data in the cloud, the TPA is trusted and its audit result is unbiased for both data owners and cloud servers; and a proxy agent, who is semi-trusted and acts on behalf of the data owner to regenerate authenticators and data blocks on the failed servers during the repair procedure. Notice that the data owner is restricted in computational and storage resources compared to other entities and may becomes off-line even after the data upload procedure. The proxy, who would always be online, is supposed to be much more powerful than the data owner but less than the cloud servers in terms of computation and memory capacity. To save resources as well as the online burden potentially brought by the periodic auditing and accidental repairing, the data owners resort to the TPA for integrity verification and delegate the reparation to the proxy. Compared with the traditional public auditing system model, our system model involves an additional proxy agent. In order to reveal the rationality of our design and make our following description to be more clear and concrete.

3.1 *Existing System*

To securely introduce an effective third party auditor (TPA), the following two fundamental requirements have to be met:
1) TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user;
2) The third party auditing process should bring in no new vulnerabilities towards user data privacy. Network codes designed specifically for distributed storage systems have the potential to provide dramatically higher storage efficiency for the same availability. One main challenge in the design of such codes is the exact repair problem: if a node storing encoded information fails, in order to maintain the same level of reliability we need to create encoded information at a new node.

3.2 Architecture Design



One of the main open problems in this emerging area has been the design of simple coding schemes that allow exact and low cost repair of failed nodes and have high data rates. The technique implemented for encrypting outsourcing data AES (acronym of Advanced Encryption Standard) is a symmetric encryption algorithm. The algorithm was developed by two Belgian cryptographer Joan Daemen and Vincent Rijmen.
AES was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits. AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field. The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext.

4. PROPOSED SYTEM DESCRIPTION

Among the first few ones to support privacy-preserving public auditing in cloud computing, with a focus on data storage. Besides, with the prevalence of cloud computing, a foreseeable increase of auditing tasks from different users may be delegated to TPA. As the individual auditing of these growing tasks can be tedious and cumbersome, a natural demand is then how to enable the TPA to efficiently perform multiple auditing tasks in a batch manner, i.e., simultaneously. The public auditing system of data storage security in cloud computing using proxy and provide a privacy-preserving auditing protocol.

5. PERFORMANCE ANALYSIS

This authenticator can be efficiently generated by the data owner simultaneously with the encoding procedure. Extensive analysis shows that our scheme is provable secure, and the performance evaluation shows that our scheme is highly efficient and can be feasibly integrated into a regenerating- code-based cloud storage system.

6. CONCLUSION AND FUTURE ENHANCEMENT

A public auditing scheme for regenerating-code-based cloud storage system, where the data owners are privileged to delegate TPA for their data validity checking. To protect the original data privacy against the TPA, we randomize the coefficients in the beginning rather than applying the blind technique during the auditing process .Considering that the data owner cannot always stay online in practice, in order to keep the storage available and verifiable after a malicious corruption, we introduce a semi-trusted proxy into the system model and provide a privilege for the proxy to handle the reparation of the coded blocks and authenticators. To better appropriate for the regenerating-code-scenario, we design our authenticator based on the BLS signature. In Future, when uploading a file, the file will be divided into number of segments (eg: A, B, C). And those segments are stored in more than one server.(eg :server 1, server 2, server 3). Server 1 containing A and B segments, server2 containing B and C segments, Server 3 containing C and A segments. If anyone server fails the file in that particular server also corrupted. Then the original file will be retrieved with the help of remaining servers. It reduces the complexity to identify the file and easily recover the original file.

7. ACKNOWLEDGMENTS

We would like to thank all the authors of different research papers referred during writing this paper. It was very knowledge gaining and helpful for the further research to be done in future. Also I would like to thank my guide Prof. Jeyanthi for her valuable guidance.

8. REFERENCES

[1] G. Ateniese et al., “Provable data possession at untrusted stores,” in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), New York, NY,USA, 2007, pp. 598–609.

[2] A. Juels and B. S. Kaliski, Jr., “PORs: Proofs of retrievability for large files,” in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 584–597.

[3] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, “MR-PDP:Multiple-replica provable data possession,” in *Proc. 28th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2008, pp. 411– 420.

[4] K. D. Bowers, A. Juels, and A. Oprea, “HAIL: A high- availability and integrity layer for cloud storage,” in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 187–198.

[5] J. He, Y. Zhang, G. Huang, Y. Shi, and J. Cao, “Distributed data possession checking for securing multiple replicas in geographically dispersed clouds,” J. Comput. Syst. Sci., vol. 78, no. 5, pp. 1345–1358, 2012.

[6] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, “Remote data checking for network coding-based distributed storage systems,” in Proc. ACM Workshop Cloud Comput. Secur. Workshop, 2010, pp. 31–42.

[7] H. C. H. Chen and P. P. C. Lee, “Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation,”IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 407–416, Feb. 2014.

[8] K. Yang and X. Jia, “An efficient and secure dynamic auditing protocol for data storage in cloud computing,” IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[9] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, “Cooperative provable data Possession for integrity verification in multicloud storage,” IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 12, pp. 2231– 2244, Dec. 2012.

[10] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, “A survey on network codes for distributed storage,” Proc. IEEE, vol. 99, no. 3, pp. 476–489, Mar. 2011.

[11] H. Shacham and B. Waters, “Compact proofs of retrievability,” in Advances in Cryptology. Berlin, Germany: Springer-Verlag, 2008, pp. 90–107.

[12] Y. Hu, H. C. H. Chen, P. P. C. Lee, and Y. Tang, “NCCloud: Applying network coding for the storage repair in a cloud-of-clouds,” in *Proc. USENIX FAST*, 2012, p. 21.

[13] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, “Dynamic provable data possession,” in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213