International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume 2, Issue 3 (March'2016)

CLOUD AUDITING USING REGENERATING CODES WITH ENCRYPTED KEYS FOR EXTENSIVE SECURITY

R.Raja,
Assistant Professor,
Department of Computer science and engineering,
Velammal Institute of Technology,Chennai,India.

V.Nishanthi,
Student,
Department of Computer science and engineering,
Velammal Institute of Technology,Chennai,India.

B.Keerthana,
Student,
Department of Computer Science and Engineering,
Velammal Institute of Technology,Chennai,India.

S.Monisha,
Student,
Department of Computer science and engineering,
Velammal Institute of Technology,Chennai,India.

Abstract: Data security and privacy are the critical issues for remote data storage. A secure user enforced data access control mechanism must be provided before cloud users have the liberty to outsource sensitive data to the cloud for storage. With the emergence of sharing confidential corporate data on cloud servers, it is imperative to adopt an efficient encryption system with a fine-grained access control to encrypt outsourced data. Existing remote checking methods for regenerating-coded data only provide private auditing, requiring data owners to always stay online and handle auditing, as well as repairing, which is sometimes impractical. In this paper, we propose a public auditing scheme for the regenerating-code-based cloud storage. To solve the regeneration problem of failed authenticators in the absence of data owners, we introduce a proxy, which is privileged to regenerate the authenticators, into the traditional public auditing system model. Attribute-based encryption is a public key based encryption that enables access control over encrypted data using access policies and ascribed attributes. Extensive security analysis shows that our scheme is provable secure under random oracle model and experimental evaluation indicates that our scheme is highly efficient and can be feasibly integrated into the regenerating- code-based cloud storage.

Keywords: Cloud storage, regenerating codes, public audit, authenticator regeneration, proxy, encrypted public key.

I. INTRODUCTION

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 maintenances, etc., Nevertheless, this new paradigm of data hosting service also brings new security threats toward users 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 signifi-cant work among these studies are the PDP (provable datapossession) model and POR (proof of retrievability) model, which were originally proposed for the **single-server** scenario by Atenieseet al. [2] and Juels and Kaliski [3], respectively. Considering that files are usually striped and redundantly stored across multi-servers or multi-clouds, [4]–[10] 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 [11]. Similar studies have been performed by Chen et al. [7] and Chen and Lee [8] separately and independently. [7] extended the single-server CPOR scheme (private version in [12]) to the regenerating-code-scenario; [8] designed and implemented a data integrity protection (DIP) scheme for FMSR [13]-based cloud storage and the scheme is adapted to the thin-cloud setting.[1]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 [14]. 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) [15]. In

IJCRCST © 2016 | All Rights Reserved

particular, users may not want to go through the complexity in verifying and reparation. The auditing schemes in [7] and [8] 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 (of failed data blocks and authenticators) 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 [12] 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 [14] and [15] and data blind method in [16]. Several challenges and threats spontaneously arise in our new system model with a proxy (Section II-C), and security analysis shows that our scheme works well with these problems. Specifically, our contribution can be summarized by the following .We design a novel homomorphic authenticator based on BLS signature [17], 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 illustrated in Section II-C .Moreover, we make a comparison with the state of the art and experimentally evaluate the performance of our scheme.

The rest of this paper is organized as follows: Section II introduces some preliminaries, the system model, threat model, design goals and formal definition of our auditing scheme. Then we provide the detailed description of our scheme in Section III; Section IV analyzes its security and Section V evaluates its performance. Section VI presents a review of the related work on the auditing schemes in cloud storage. Finally, we conclude this paper in Section VII.

II. RELATED WORK

[1] "Provable data possession at untrusted stores, From This Paper we Referred-

This keynote paper: In Cloud Computing moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This paper addressed the problem of ensuring the integrity of data storage in Cloud Computing.

[2] PORs: Proofs of Retrievability for large files From This Paper we Referred-

The distributed storage systems apply redundancy coding techniques to stored data. One form of redundancy is based on regenerating codes, which can minimize the repair bandwidth, i.e., the amount of data transferred when repairing a failed storage node. Existing regenerating codes mainly require surviving storage nodes encode data during repair.

[3] Multiple-replica provable data possession From This Paper we Referred-

In this approach, cloud computing is to avail all the resources at one place in the form a cluster and to perform the resource allocation based on request performed by different users. They defined the user request in the form of requirement query. Cloud Computing devices being able to exchange data such as text files as well as business information with the help of internet. Technically, it is completely distinct from an infrared. Using new models Iaas, Paas and Saas.

A. Notations and Preliminaries

1) Regenerating Codes: Regenerating codes are first introduced by Dimakiset al. [18] for distributed storage to reduce the repair bandwidth.When data corruption at a server is detected, the client will contact healthy servers and download β- bits from each server, thus regener-ating the corrupted blocks without recovering the entire original file. Dimakiset al. [18] showed that the repair bandwidth γ-=_β- can be significantly reduced with _≥k. Furthermore, they analyzed the fundamental tradeoff between the storage cost α- and the repair bandwidth γ- , then presented two extreme and practically relevant points on the optimal tradeoff curve: the minimum bandwidth regenerat-ing(MBR) point, which represents the operating point with the least possible repair bandwidth, and the minimum storage regenerating (MSR) point, which corresponds to the least possible storage cost on the servers. Denoted by the parameter tuple (n,k,_, α-,

γ-) , we obtain.

Moreover, according to whether the corrupted blocks can be exactly regenerated, there are two versions of repair strategy: exact repair and functional repair [11]. Exact repair strategy requires the repaired server to store an exact replica of the corrupted blocks, while functional repair indicates that the newly generated blocks are different from the corrupted ones with high probability. As one basis of our work, the functional repair regenerating codes are non-systematic and do not perform as well for read operation as systematic codes, but they really make sense for the scenario in which data repair occurs much more often than read, such as regulatory storage, data escrow and long-term archival storage [7].

The finite field G F (2^{ω}) , where the coefficients are drawn, is large enough, any m coded blocks will be linearly independent with high probability [19]; thus, the original file can be reconstructed from any m coded blocks by solving a set of m equations. The coded blocks will be distributed across n storage servers, with each server storing blocks. Any k-out-of-n servers can be used to recover the original data file when the inequality km is maintained, thus satisfying the MDS property. When the periodic auditing process detect data corruption at one server, the repair procedure will contact _ servers and obtain β blocks from each to regenerate. The proxy agent in our system model is assumed to be semi-trusted. It will not collude with the servers but might attempt to forge authenticators for some specified invalid blocks to pass the following verification.

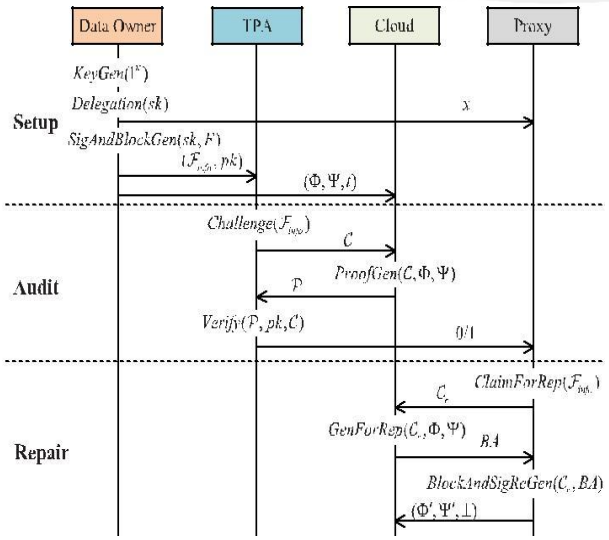

Figure 1:Detection of encrypted key.

Claim For Rep $(F_{info}) \rightarrow (C_r)$: This algorithm is similar with the Challenge () algorithm in the Audit phase, but outputs a claim for repair C_r . Gen For Rep $(C_r, ,) \rightarrow ($ B A$)$: The cloud servers run this algorithm upon receiving the C_r and finally output the block and authenticators set B A with another two inputs , B A from each server as input, and outputs a new coded blockset – and authenticator set – if successful, outputting ⊥ if otherwise.

III. THE PROPOSED SYSTEM.

Thus, a problem arises when trying to determine how to regenerate authenticators for the repaired blocks. A direct solution, which is adopted in [7], is to make data owners handle the regeneration. However, this solution is not practical because the data owners will not always remain online through the life-cycle of their data in the cloud, more typically, it becomes off-line even after data uploading. Another challenge is brought in by the proxy in our system model (see Section II-C).

The following parts of this section shows our solution to the problems above. First, we construct a BLS-based [17] authenticator, which consists of two parts for each segment of coded blocks. Utilizing its homomorphic property and the linearity relation amongst the coded blocks, the data owner is able to generate those authenticators in a new method, which is more efficient compared to the straightforward approach. Our authenticator contains the information of encoding coefficients to avoid data pollution in the reparation with wrong coefficients. To reduce the bandwidth cost during the audit phase, we perform a batch verification over all α blocks at a certain server rather than checking the integrity of each block separately as [7] does. Moreover, to make our scheme secure against the replace attack and replay attack, information about the indexes of the server, blocks, and segments are all embedded into the authenticator. Besides, our primitive scheme can be easily improved to support privacy-preserving through the masking of the coding coefficients with a keyed PRF. All the algebraic operations in our scheme work over the field G F (p) of modulo p, where p is a large prime (at least 80 bits). In this section we start from an overview of our auditing scheme, and then describe the main scheme and discuss how to generalize our privacy-preserving public auditing scheme. Furthermore, we illustrate some optimized methods to improve its performance.

Advantages of Proposed System

Public Auditability:To allow TPA to verify the intactness of the data in the cloud on demand without introducing additional online burden to the data owner.

Storage Soundness: to ensure that the cloud server can never pass the auditing procedure except when it indeed manages the owner's data intact.

Privacy Preserving: to ensure that neither the auditor nor the proxy can derive users' data content from the auditing and reparation process.

Authenticator Regeneration: the authenticator of the re-paired blocks can be correctly regenerated in the absence of

the data owner.

Error Location: to ensure that the wrong server can be quickly indicated when data corruption is detected.

IV. SECURITY ANALYSIS

In this section, we first elaborate on the correctness of verification in our auditing scheme and then formally evaluate the security by analyzing its fulfillment of soundness, regeneration-unforgeable and secure guarantee against replay attack.

A. Correctness.

There are two verification process in our scheme, one for spot checking during the **Audit** phase and another for block integrity checking during the **Repair**phase.Theorem 1: Given a cloud server i storing data blocks $_{i}$ and accompanied authenticators $_{i}$, TPA is able to correctly verify its possession of those data blocks during audit phase, and the proxy can correctly check the integrity of downloaded blocks during repair phase.

Proof: Proving the correctness of our auditing scheme is equivalent of proving that Eq.(15) and Eq.(17) is correct. The correctness of Eq.(15) is shown in Appendix A and Eq.(17) in Appendix B.

B. Soundness.

Following from paper [12], we say that our auditing protocol is sound if any cheating server that convinces the verification algorithm that it is storing the coded blocks and corresponding coefficients is actually storing them. Before proving the soundness, we will first show that the authenticator as Eq.(10) is unforgeable against malicious cloud servers (referred to as adversary in the following definition) under the random oracle model.

V. PERFORMANCE ANALYSIS:

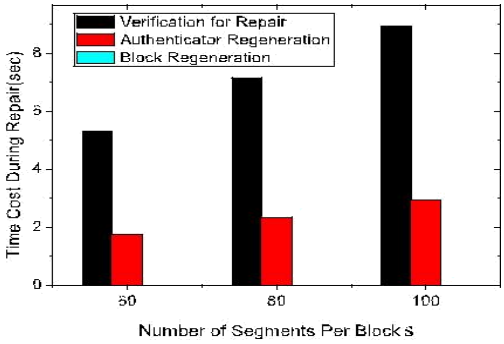

Figure 2: Time for Repair with differents

It takes the proxy much more time to verify the received blocks for repair, less to regenerate the authenticators, and negligible to regenerate the faulty blocks. This situation occurs mainly because there are a mass of expensive pairing operations and less expensive modular exponentiations during the verification, thus leading to the most time consuming sub-process. In contrast, there are only asymptotic αs (_+α_+1) modular exponentiations and αs inversions in a finite group across the authenticator regeneration, and only αs multiplications during the block regeneration. However,the efficiency of the verification can be significantly improved using a batch method, where we can randomly choose s weights for each received blocks, aggregate the segments and authenticators, and then verify them all in one, thus we can avoid applying so many pairing operations.

CONCLUSION

The main access polices are KP-ABE and CP-ABE, further schemes are obtained based on these policies. In this paper, we propose a privacy-preserving public auditing system for data storage security in Cloud Computing. We utilize the homomorphic linear authenticator and random masking to guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. .Extensive security analysis shows that our scheme is provable secure under random oracle model and experimental evaluation indicates that our scheme is highly efficient and can be feasibly integrated into the regenerating- code-based cloud storage.

REFERENCES:

[1] M. Armbrustet al., "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

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

[3] 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.

[4] 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.

[5] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in Proc. 16th ACM Conf. Comput..