Regenerating Code Based Privacy Preserving Public Auditing For Secure Cloud Storage

I.Miltan Prabhakaran and S.Dharanitharan, B.Tech(ME), Assistant Professor, Department of CSE, JKK Munirajah College of Technology, Gobi, India

Abstract - Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a third party auditor (TPA) to check the integrity of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user. In this paper, propose a secure cloud storage system supporting privacy-preserving public auditing with proxy server which is regenerate the code for data recovery.

Keywords: Data Storage, Privacy Preserving, Public Auditability, Cloud Computing, TPA, Proxy.

I. INTRODUCTION

Cloud computing has been envisioned as the next generation information technology architecture for enterprises, due to its long list of unprecedented advantages in the IT history on demand self service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage based pricing and transference of risk. As a disruptive technology [7] with profound implications, cloud computing is transforming the very nature of how businesses use information technology. One fundamental aspect of this paradigm shifting is that data are being centralized or outsourced to the cloud [2]. From user’s perspective, including both individuals and IT enterprises, storing data remotely to the cloud in a flexible on demand manner brings appealing benefits relief of the burden for storage management, universal data access with location independence, and avoidance of capital expenditure on hardware, software, and personnel maintenance, etc... Advantages of this are Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network [13]. The cloud also focuses on maximizing the effectiveness of the shared resources, Cloud resources are usually not only shared by multiple users but are also dynamically re allocated per demand. This approach should maximize the use of computing powers thus reducing environmental damage as well since less power, air conditioning, rack space, etc. is required for a variety of functions.

II. LITERATURE REVIEW & RELATED WORK

A. The System and Threat Model

We consider a cloud data storage service involving three different entities, as illustrated in the cloud user (U) [12], who has large amount of data files to be stored in the cloud; the cloud server (CS), which is managed by the cloud service provider (CSP) [3] to provide data storage service and has significant storage space and computation resources (we will not differentiate CS and CSP hereafter); the third party auditor (TPA), who has expertise and capabilities that cloud users do not have and is trusted [6] to assess the cloud storage service reliability on behalf of the user upon request. Users rely on the CS for cloud data storage [9] and maintenance. They may also dynamically interact with the CS to access and update their stored data for various application purposes. To save the computation resource as well as the online burden, [11] cloud users may resort to TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA.

We consider the existence of a semi-trusted CS as does. Namely, in most of time it behaves properly and does not deviate from the prescribed protocol execution. However, 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 to hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation. We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. However, it harms the user if the TPA could learn the outsourced data after the audit.

To authorize the CS to respond to the audit delegated to TPA’s [6], the user can sign a certificate granting audit rights to the TPA’s public key, and all audits from the TPA are authenticated against such a certificate. These authentication handshakes are omitted in the following presentation.

B. Design Goals

To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantees [8].

Fig 1: Architecture of Cloud Storage.
1) Public auditability: to allow TPA to verify the correctness [4] of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the cloud users.

2) Storage correctness: to ensure that there exists no cheating cloud server that can pass the TPA’s audit without indeed storing users’ data intact.

3) Privacy-preserving: to ensure that the TPA cannot derive users’ data content from the information collected during the auditing process.

4) Batch auditing: to enable TPA [2] with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different users simultaneously.

5) Lightweight: to allow TPA to perform auditing with minimum communication and computation overhead.

The primary improvements are as follows: To protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage together with data integrity checking and failure reparation becomes critical. Recently, regenerating codes have gained popularity due to their lower repair bandwidth while providing fault tolerance [7]. 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 [9] 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. Moreover, we design a novel public verifiable authenticator [10], which is generated by a couple of keys and can be regenerated using partial keys. Thus, our scheme can completely release data owners from online burden. In addition, we randomize the encode coefficients with a pseudorandom function to preserve data privacy. Extensive security analysis shows that our scheme is provably 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.

III. THE PROPOSED SCHEME

This section our public auditing scheme for the 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 [6] 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. This authenticator can be efficiently generated by the data owner simultaneously with the encoding procedure. Extensive analysis shows that our scheme is provably secure [8], and the performance evaluation shows that our scheme is highly efficient [9] and can be feasibly integrated into a regenerating-code-based cloud storage system.

Specifically, our contribution can be summarized by the following aspects:

We design a novel homomorphic authenticator based on BLS signature [7], 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 light weight 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 [11].

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.

A. Notations and Preliminaries

Regenerating Codes: Regenerating codes are first introduced by A. G. Demakis et al. for distributed storage to reduce the repair bandwidth. Viewing cloud storage to be a collection of n storage servers, data file F is encoded and stored redundantly across these servers. Moreover, according to whether the corrupted blocks can be exactly regenerated, there are two versions of repair strategy: exact repair and functional repair. Exact repair strategy [11] 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 [9], 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.

B. System Model

We consider the auditing system model for Regenerating Code-based cloud storage as which involves four entities: the data owner, who owns large amounts of data files to be stored in the cloud [6]; the cloud, which are managed by the cloud service provider, provide storage service and have significant computational resources; the third party auditor(TPA) [6], 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 [9]. 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 [7], 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 [13].

\[ \text{Setup}. \] The data owner maintains this procedure to initialize the auditing scheme.

\[ \text{KeyGen}(1^k) \to (pk, sk): \] This polynomial-time algorithm is run by the data owner to initialize its public and secret parameters by taking a security parameter $k$ as input.

\[ \text{Degelation}(sk) \to (x): \] This algorithm represents the interaction between the data owner and proxy. The data owner delivers the partial secret key $x$ to the proxy through a secure approach.

\[ \text{SigAndBlockGen}(sk, F) \to (\Phi, \Psi, t): \] This polynomial time algorithm is run by the data owner and takes the secret parameter $sk$ and the original file $F$ as input, and then outputs a coded block set $\Psi$, an authenticator set $\Phi$ and a file tag $t$.

\[ \text{Audit}. \] The cloud servers and TPA interact with one another to take a random sample on the blocks and check the data integrity in this procedure.

\[ \text{Challenge}(\text{Fin}\, o) \to (C): \] This algorithm is performed by the TPA with the information of the file $F$ info as input and a challenge $C$ as output.

\[ \text{ProofGen}(C, \Phi, \Psi \to (P)): \] This algorithm is run by each cloud server with input challenge $C$, coded block set $\Psi$ and authenticator set $\Phi$, then it outputs a proof $P$.

\[ \text{Verify}(P, pk, C) \to (0, 1): \] This algorithm is run by TPA immediately after a proof is received. Taking the proof $P$, public parameter $pk$ and the corresponding challenge $C$ as input, it outputs 1 if the verification passed and 0 otherwise.

\[ \text{Repair}. \] In the absence of the data owner, the proxy interacts with the cloud servers during this procedure to repair the wrong server detected by the auditing process.

**CONCLUSION**

In this paper, we propose a public auditing scheme for the 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. This authenticator can be efficiently generated by the data owner simultaneously with the encoding procedure. Extensive analysis shows that our scheme is provably 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.

**References**


[11] Shared Authority Based Privacy-preserving Authentication Protocol in Cloud Computing Hong Liu, Student Member, IEEE, Huansheng Ning, Senior Member, IEEE, Qingxu Xiong, Member, IEEE and Laurence T. Yang, Member, IEEE.


BIOGRAPHIES


Name: Mr. Dharanitharan. S, Qualification: M.E., MISTE, Designation: AP (CSE) JKKM College of Technology, Gobi, Erode, India. Area of Interest: DBMS, Digital Signal Processing.