A New Delegating Auditing Task to TPA for Storage Correctness and Privacy in Cloud

Lakshmanarao Simhadri [#1], Rajendra Kumar Ganiya [*2]

M.Tech Scholar [#1], Professor & HOD [*2]

Department of Computer Science & Engineering,
Sri Sivani College of Engineering, Chilkapalem,
Etcherla Mandal, Srikakulam District.532402.

Abstract

Cloud Computing is one of the recent attraction in almost all types of business environments, where it is used for storing a lot of individual private data on to a remote systems. As the data is always stored remotely, we can't able to give guarantee whether the data was safe or it have been misused. Cloud means collection of storage servers maintained by the cloud service provider which minimizes investment cost for individual users and organizations. It providing on-demand self-service, resource pooling, rapid elasticity and measured service. But users are worrying about their data stored in untrusted cloud servers. For that introducing third-party auditor along with privacy preserving public auditing technique which audit, verifies and provides privacy of user's data in cloud. In this paper, we propose a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient.

Keywords

Cloud Computing, Data Storage, Privacy-Preserving, Public Auditability, Cryptographic Protocols

1. Introduction

Cloud storage denotes a family of increasingly popular on-line services for archiving, backup, and even primary storage of files. Amazon S3 is a well-known example. Cloud-storage providers offer users clean and simple file-system interfaces, abstracting away the complexities of direct hardware management. As a standalone tool for testing file retrievability against a single server, though, a POR is of limited value.1 Detecting that a file is corrupted is not helpful if the file is irretrievable and thus the client has no recourse. Thus PORs are mainly useful in environments where *F* is distributed across multiple systems, such as independent storage services. A POR uses file redundancy *within a server* for verification. In a second, complementary approach, researchers have proposed distributed protocols that rely on queries *across servers* to check file availability.

Strong File-Intactness Assurance:

HAIL enables a set of servers to prove to a client via a challenge-response protocol that a stored file *F* is fully intact—more precisely, that the client can recover *F* with overwhelming probability.

Low Overhead:

The per-server computation and bandwidth required for HAIL is comparable to that of previously proposed PORs. Apart from its use of a natural file sharing across servers, HAIL improves on PORs by eliminating check values and reducing within-server file expansion

Strong Adversarial Model:

HAIL protects against an adversary that is *active*, i.e., can corrupt servers and alter file blocks and *mobile*, i.e., can corrupt every server over time.

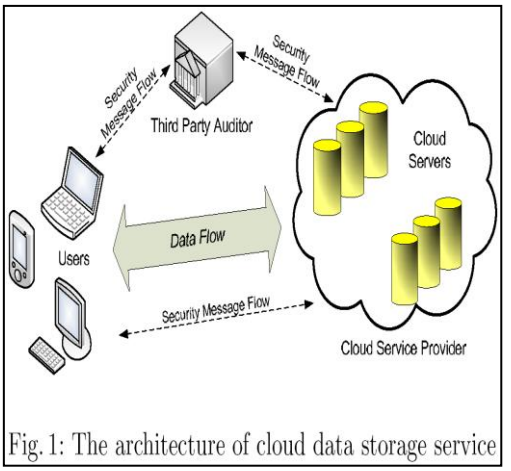

Fig. 1: The architecture of cloud data storage service

2. Related Work

In this section we will describe the assumptions and background knowledge that is used for developing the new auditing method for cloud data storage.

2.1 The System Model

Ateniese et al. [1] are the first to consider public auditability in their "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the RSA-based homomorphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. However, among their two proposed schemes, the one with public auditability exposes the linear combination of sampled blocks to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the external auditor. Juels et al. [2] describe a "proof of retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on remote archive service systems. However, the number of audit challenges a user can perform is fixed a priori, and public auditability is not supported in their main scheme. Although they describe a straightforward Merkle-tree construction for public PoRs, this approach only works with encrypted data. Later, Bowers et al. [3] propose an improved framework for POR protocols that generalizes Juels' work. Dodis et al. [4] also give a study on different variants of PoR with private auditability. Shacham and Waters [5] design an improved PoR scheme built from BLS signatures [6] with proofs of security in the security model defined in [7]. Similar to the construction in [8], they use publicly verifiable homomorphic linear authenticators that are built from provably secure BLS signatures. Based on the elegant BLS construction, a compact and public verifiable scheme is obtained.

In other related work, Sebe et al. [9] thoroughly study a set of requirements which ought to be satisfied for a remote data possession checking protocol to be of practical use. Their proposed protocol supports unlimited times of file integrity verifications and allows preset tradeoff between the protocol running time and the local storage burden at the user.

2.2 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.

1) Public Auditability: to allow TPA to verify the correctness 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 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.

3. Proposed Cloud Schemes

This section presents our public auditing scheme which provides a *complete outsourcing* solution of data – not only the data itself, but also its integrity checking. We start from an overview of our public auditing system and discuss two straightforward schemes and their demerits. Then we present our main scheme and show how to extent our main scheme to support batch auditing for the TPA upon delegations from multiple users.

3.1 Definition

A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof). KeyGen is a key generation algorithm that is run by the user to setup the scheme. SigGen is used by the user to generate verification metadata, which may consist of MAC, signatures, or other related information that will be used for auditing. GenProof is run by the cloud server to generate a proof of data storage correctness, while VerifyProof is run by the TPA to audit the proof from the cloud server.

Running a public auditing system consists of two phases, Setup and Audit:

A) Setup Phase:

The user initializes the public and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SigGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and deletes its local copy. As part of pre-processing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.

B) Audit Phase:

The TPA issues an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message from a function of the stored data file F and its verification metadata by executing GenProof. The TPA then verifies the response via VerifyProof.

Our framework assumes the TPA is stateless, which is a desirable property achieved by our proposed solution. It is easy to extend the framework above to capture a stateful auditing system, essentially by spliting the verification metadata into two parts which are stored by the TPA and the cloud server respectively. Our design does not assume any additional property on the data file. If the user wants to have more error-resiliency, he/she can always first redundantly encodes the data file and then uses our system with the data file that has error-correcting codes integrated.

3.2 Detailed Explanation of Algorithm Steps

The following are the main steps of algorithm that are explained below:

1. The user blinds each file block data before file distribution k is the secret key for data vector is generated.

2. Based on the blinded data vector, the User generates k parity vector via the secret matrix P.

3. The user calculates the ith token for server j.

4. The user sends the token secret matrix P, permutation and challenge key K_{master} key, and k_{chal} to TPA for auditing delegation.

The blinding values in the servers are not taken by TPA response of the server are verified directly. As TPA does not know the secret blinding key there is no way for TPA to learn the data content information during auditing process. Thus the privacy-preserving third party auditing is achieved.

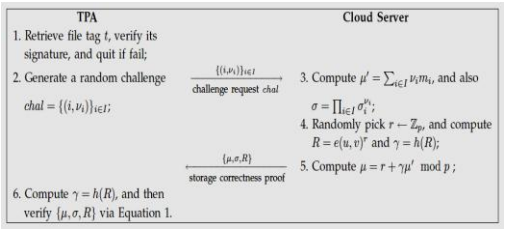

Fig.2: The privacy-preserving public auditing protocol

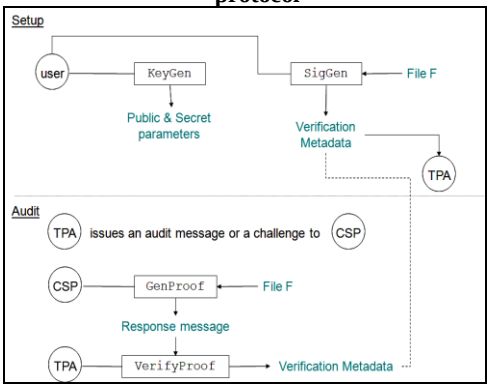

Fig.3: The Setup and Audit Phase in proposed Model

4. Implementation Modules

Implementation is the stage where the theoretical design is automatically converted into practically by dividing this into various modules. We have implemented the current application in Java Programming language with JEE as the main interface for developing the proposed application with Front End as HTML, JSP Pages and Back end as MY SQL data base for storing and retrieving the records. Our proposed application is divided into following 4 modules. They are as follows:

a) Public audit ability for storage correctness assurance
b) Dynamic data operation support
c) Block less verification
d) Dynamic Data Operation with Integrity Assurance
e) Data Modification
f) Batch Auditing for Multi-client Data:

a) Public audit ability for storage correctness assurance

In this module, to allow anyone, the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.

b) Dynamic data operation support

To allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support.

c) Blockless Verification

No challenged file blocks should be retrieved by the verifier (*e.g.*, TPA) during verification process for efficiency concern.

d) Dynamic Data Operation with Integrity Assurance

Now we show how our scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. Note that in the following descriptions, we assume that the file F and the signature _ have already been generated and properly stored at server. The root metadata R has been signed by the client and stored at the cloud server, so that anyone who has the client's public key can challenge the correctness of data storage.

e) Data Modification

We start from data modification, which is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones. At start, based on the new block the client generates the corresponding signature. The client signs the new root metadata R′ by sigsk(H(R′)) and sends it to the server for update. Finally, the client executes the default integrity verification protocol. If the Output is *TRUE*, delete sigsk(H(R′)),and generate duplicate file.

f) Batch Auditing for Multi-client Data

As cloud servers may concurrently handle multiple verification sessions from different clients, given K signatures on K distinct data files from K clients, it is more advantageous to aggregate all these signatures into a single short one and verify it at one time. To achieve this goal, we extend our scheme to allow for provable data updates and verification in a multi-client system. The signature scheme allows the creation of signatures on arbitrary distinct messages. Moreover, it supports the aggregation of multiple signatures by distinct signers on distinct messages into a single short signature, and thus greatly reduces the communication cost while providing efficient verification for the authenticity of all messages.

5. Conclusion

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. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files, we further extend our privacy-preserving public auditing protocol into a multi-user setting, where the TPA can perform multiple auditing tasks in a batch manner for better efficiency. Extensive analysis shows that our schemes are provably secure and highly efficient.

6. References

[1] 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.

[2] A. Juels and J. Burton, S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[3] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 43-54, 2009.

[4] Y. Dodis, S.P. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Theory of Cryptography Conf. Theory of Cryptography (TCC), pp. 109-127, 2009.

[5] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. Int'l Conf. Theory and Application

of Cryptology and Information Security: Advances in Cryptology (Asiacrypt), vol. 5350, pp. 90-107, Dec. 2008.

[6] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," J. Cryptology, vol. 17, no. 4, pp. 297-319, 2004.

[7] P. Mell and T. Grance, "Draft NIST Working Definition of Cloud Computing," http://csrc.nist.gov/groups/SNS/cloudcomputing/ index.html, June 2009.

[8] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010.

[9] F. Sebe, J. Domingo-Ferrer, A. Martı´nez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient Remote Data Possession Checking in Critical Information Infrastructures," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 8, pp. 1034-1038, Aug. 2008.

7. About the Authors

Lakshmanarao Simhadri is currently pursuing his 2 Years M.Tech (CSE) in Computer Science and Engineering at Sri Sivani College of Engineering, Chilkapalem, Etcherla Mandal, Srikakulam District. His area of interests includes Networks and Cloud Computing.

Rajendra Kumar Ganiya is currently working as a Professor and Head of Department with Dept. of CSE at Sri Sivani College of Engineering, Chilkapalem, Etcherla Mandal, Srikakulam District. His research interests include Cognitive Science.

