

Secured Dynamic Auditing In Cloud Storage Using Fully Homomorphic Mechanism

Author: Mr.S.Suresh Kumar¹; Mr.M.Newlin Rajkumar²

PG Scholar, Department of CSE, Anna University Regional Centre, Coimbatore¹;

Assistant professor, Department of CSE, Anna University Regional Centre, Coimbatore²

ABSTRACT

In cloud computing, data owners host their data on cloud servers and users can access the data from cloud servers. Due to the data outsourcing, however, this new paradigm of data hosting service also introduces new security challenges, which requires an independent auditing service to check the data integrity in the cloud. In existing system is desired to convince data owners that the data are correctly stored in the cloud. In that system there is no security is provided when outsourcing data from TPA to server and does not concern is how to ensure the integrity of the outsourced data. So our proposed system is mainly concentrated on provide security between TPA to server and also we extend our system to check the integrity of failure in that file using Erasure Code technique. The original auditing protocol is vulnerable to the attack from an active adversary since it does not provide authentication of the response, so we suggest employing a secure digital signature scheme to prevent the proof from being modified. We present a novel family of erasure codes that are efficiently repairable and offer higher reliability. In this work a flexible distributed storage integrity auditing mechanism, utilizing the distributed erasure-coded data. The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server.

Index terms

Auditing service, Data integrity, Third party auditor, Auditing protocol.

1. INTRODUCTION

Cloud storage allows data owners to remotely store their data and access them via networks at anytime and from anywhere. Which enables benefits like, relief from storage management for huge data, universally data can be accessed, reduced hardware, software requirements and maintenance. Despite the obvious benefits such as improved scalability and accessibility, data replication and backup of cloud

storage, it also brings new security challenges to the cloud data security. Once the data are outsourced, the data owners relinquish the control over the fate of their data. The server may hide data loss accidents to maintain the reputation, or discard the data which have not been or are rarely accessed to save storage space. Therefore, it is highly essential for data owners to check the integrity and availability of the cloud data.

In cloud storage system, however, it is inappropriate to let either side of cloud service providers or owners conduct such auditing, because none of them could be guaranteed to provide unbiased auditing result. In this situation, third-party auditing is a natural choice for the storage auditing in cloud computing. A third party auditor (auditor) that has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners. For the third-party auditing in cloud storage systems, there are several important requirements that have been proposed. The auditing protocol should have the following properties: 1) Confidentiality. The auditing protocol should keep owner's data confidential against the auditor. 2) Dynamic auditing. The auditing protocol should support the dynamic updates of the data in the cloud. 3) Batch auditing. The auditing protocol should also be able to support the batch auditing for multiple owners and multiple clouds.

As data owners host their valuable data on remote storage, in traditional mechanism they protected data by preventing unauthorised access of storage services, this restricts people accessing and data updating process. several mechanisms has been proposed for verifying integrity of data stored in remote server, in which entire data has to be downloaded, this involves increase communication cost. For complete trust of integrity of data and reducing computation cost, communication and enabling public auditability of data, users may depends on independent Third party auditor (TPA).TPA audit the data stored in cloud server on behalf of user and also periodically it verify integrity of all the data stored .This mechanism can be the easiest way of finding correctness of data in cloud.

In this auditing protocol employing a fully homomorphic authenticator, PA does not require any local copy of data for verifying integrity. This reduces computation cost and number of communication over the internet. By applying masking technique together with authenticator this makes TPA unable gain the information about the data stored in cloud. And also adding dynamic audit services for verifying integrity of outsourced data. Our supports dynamic operations such as update delete and insert with timely detection of abnormal activities.

2. BACKGROUND AND RELEATED WORK

The traditional cryptographic techniques for verifying integrity and availability of data are done by using hash functions and signatures. These schemes are not applicable to outsourced data, without local copy it cannot produce any results. Audit service by TPA is more significant because of its digital forensics and trustworthiness over data in cloud. To implement public audit ability, proof of retrievability and proof of data possession have proposed by researchers. It is a probabilistic proof technique; by using this storage provider can prove data stored remains original. POR/POD schemes work on auditing protocol, so that anyone can publically verify availability and integrity of data.

In earlier auditing schemes for outsourced data, content based comparison were used to check data but it wont acceptable for irregular data types. After several architectures for public verifiable audit services has been constructed , in which uploaded data is divided into k blocks in turn each block may divided into sectors and verification tags are generated for each block. Each block must be fixed and equal size. For example a data of 524 KB can be divided into blocks of 20byteseach; approximately 25k blocks are generated along with tag. Huge amount of storage will be required for storing those tags. The additional storage required by system makes reduced efficiency.

Several factors has to be consider on design privacy preserving auditing protocol (that is protocol should protect privacy of data against the auditor). Auditor can obtain data from the proof information generated for the particular block. In case of encrypted data, high probability of breaking cipher text on knowing of secretes keys. To overcome this problem, encrypted proof information can be used along with time stamp which runs on bilinear property of bilinear paring, so that block information cannot be retrieved.

Auditor is capable of computing auditing information and auditor has to conduct auditing for several servers or same user may give request to audit their own data which stored on several servers. on increasing number of request auditor may get overloaded or bottleneck occurs. This can be avoided by running auditing service in remote cloud server independently. Server compute intermediate information based on challenge and combinations of block request received from auditor. TPA receives calculated prove information to verify originality of

data. This technique involves only less communication and computation cost by transferring load to cloud server.

3. PROBLEM STATEMENT

There is a security flaw when an active adversary is involved in the protocol. Specifically, an active adversary is able to arbitrarily modify the cloud data and produce a valid auditing proof to pass the auditing verification. As a result, this adversary can fool the auditor and the data owner to believe that the data are well maintained in the cloud, while the data has already been corrupted. In that homomorphic encryption scheme not even semantically secure may also have stronger attacks on their one-wayness. Finally, there are schemes that use a singly homomorphic encryption scheme to construct a scheme that can perform more complicated homomorphic operations. Therefore, although outsourcing data into the cloud is economically attractive for the constant complexity of long-term large-scale data storage, it slacking of offering strong assurance of data integrity and availability may impede its wide adoption by both enterprise and individual cloud users.

4. OVERVIEW OF SOLUTION

Our proposed system desirable feature of data privacy, and can be extended to support dynamic auditing and batch auditing for multiple owners and multiple clouds. It is easy to verify that the fixed protocol still preserves the properties of the original protocol such as dynamic auditing and batch auditing. For the performance of the fixed protocol, it is slightly heavier in computation and communication than the original protocol, since the server needs to compute a signature σ and forward it to the auditor additionally, and the auditor will perform extra signature verification. We propose the fully homomorphic encryption scheme, solving a central open problem in cryptography. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key. More broadly, fully homomorphic encryption improves the efficiency of secure multiparty computation. Finally In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed.

5. SYSTEM ARCHITECTURE

We introduce architecture for verifying integrity and availability of data stored in clouds as shown in fig 1.1. In this architecture we consider three entities: data owner who has large amount of data to be stored in the cloud, cloud service provider who provide data storage service and computation resources. Third party auditor (TPA) who has capabilities to manage or monitor outsourced data under the delegation of data owner. TPA regularly checks the correctness and availability of data stored on the cloud.

Key Generation Algorithm: This key generation algorithm takes no input other than the implicit

security parameter lambda. It outputs a secret hash

key s_{kh} and a pair of secret-public tag key (s_{kt}, p_{kt}) .

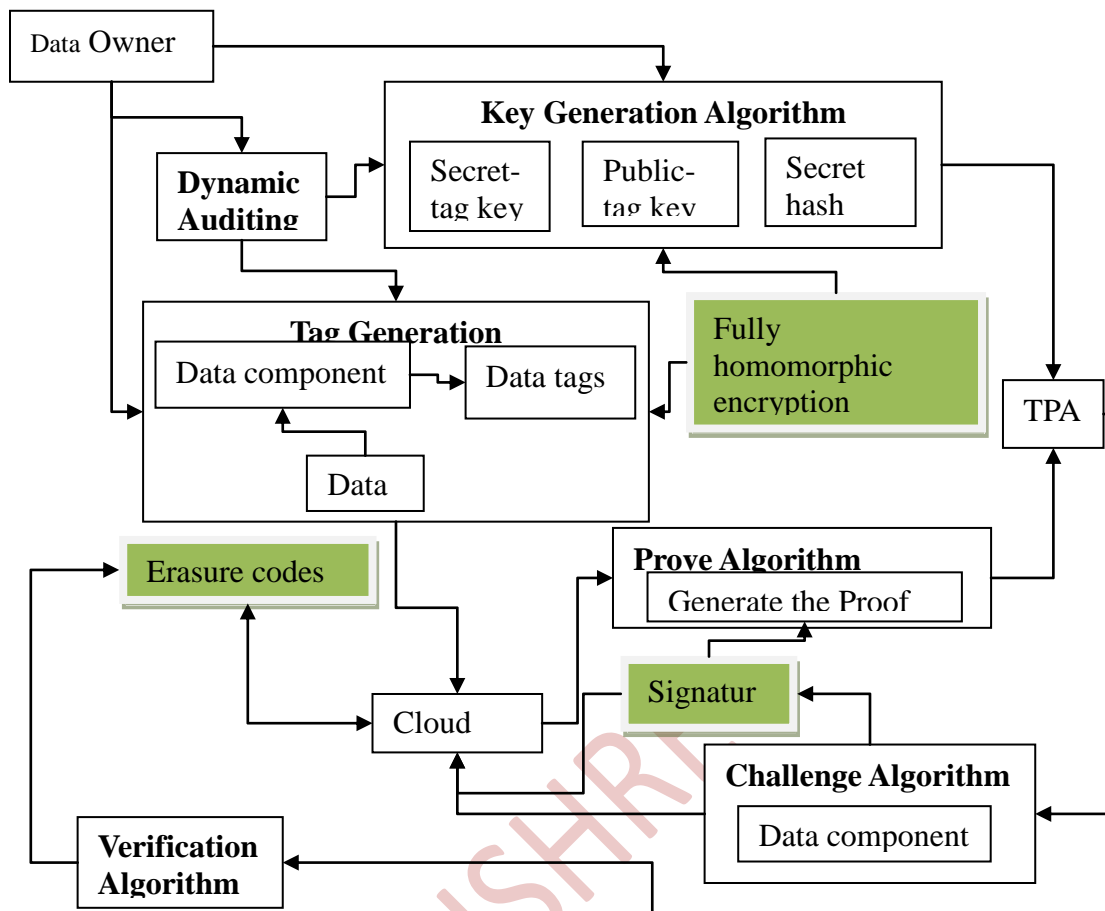


Fig 1: SYSTEM ARCHICTECTURE

Tag Generation Algorithm: The tag generation algorithm takes as inputs an encrypted file M , the secret tagkey s_{kt} , and the secret hash key s_{kh} . For each data block m_i , it computes a data tag t_i based on s_{kh} and s_{kt} .

Challenge algorithm: The challenge algorithm takes as input the abstract information of the data M_{info} (e.g., file identity, total number of blocks, version number, time stamp, etc.). It outputs a challenge C .

Prove Algorithm: The prove algorithm takes as inputs the file M , the tags T , and the challenge from the auditor C . It outputs a proof P .

Verify Algorithm: The verification algorithm takes as inputs P from the server, the secret hash key s_{kh} , the public tag key p_{kt} , and the abstract information of the data D_{info} . It outputs the auditing result.

6. AUDITING MECHANISM

Three different network entities can be identified as follows: User: an entity, who has data to be stored in the cloud and relies on the cloud for data storage and computation, can be either enterprise or individual customers. Cloud Server (CS): an entity, which is managed by cloud service provider (CSP) to provide data storage service and has significant storage space

and computation resources (we will not differentiate CS and CSP hereafter). Third-Party Auditor: an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

6.1 Initialization Process

Figure 2 shows various phases in storage auditing protocol: owner initialization, confirmation auditing, and sampling auditing.

Owner Initialization: The owner runs the key generation algorithm Key Gen to generate the secret hash key s_{kh} , the pair of secret-public tag key (s_{kt}, p_{kt}) . Then, it runs the tag generation algorithm TagGen to compute the data tags. After all the data tags are generated, the owner sends each data component $D = \{d_i\}_{i \in [1,n]}$ and its corresponding data tags $T = \{t_i\}_{i \in [1,n]}$ to the server together with the set of parameters $\{u_j\}_{j \in [1,s]}$. The owner then sends the public tag key p_{kt} , the secret hash key s_{kh} , and the abstract information of the data D_{info} to the auditor, which includes the data identifier FID, the total number of data blocks n .

Confirmation auditing: In our auditing construction, the auditing protocol only involves two-way communication:

Challenge and Proof: During the confirmation auditing phase, the owner requires the auditor to check whether the owner’s data is correctly stored on the server. The auditor conducts the confirmation auditing phase as

1. The auditor runs the challenge algorithm Chall to generate the challenge C for all the data blocks in the data component and sends the $C = (\{i, v_i\}_{i \in Q}, R)$ to the server.
2. Upon receiving the challenge C from the auditor, the server runs the prove algorithm Prove to generate the proof P = (TP, DP) and sends it back to the auditor.

3. When the auditor receives the proof from the server, it runs the verification algorithm Verify to check the correctness of P and extract the auditing result.

The auditor then sends the auditing result to the owner. If the result is true, the owner is convinced that its data are correctly stored on the server, and it may choose to delete the local version of the data.

Sampling auditing: The auditor will carry out the sampling auditing periodically by challenging a sample set of data blocks. The frequency of taking auditing operation depends on the service agreement between the data owner and the auditor (and also depends on how much trust the data owner has over the server).

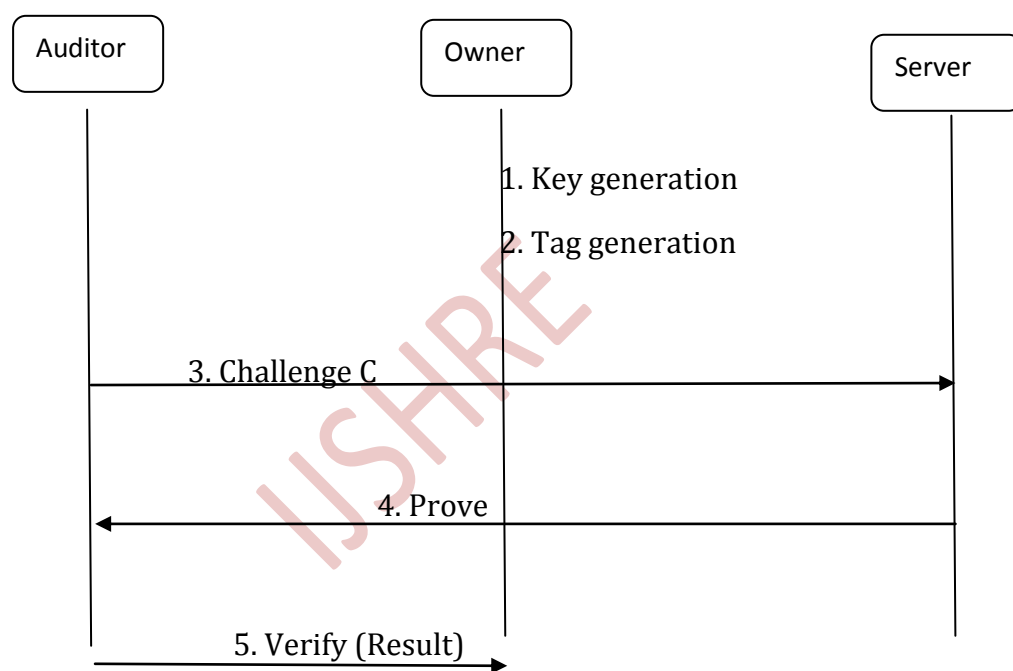


Fig 2:Data Flow Diagram

6.2 Dynamic Operation for Cloud User

Dynamic operation for cloud user is depicted in Figure 5.2. Which contains three steps: data update, index update, and update confirmation.

Step 1: Data update. There are three types of data update operations that can be used by the owner: modification, insertion, and deletion. For each update operation, there is a corresponding algorithm in the dynamic auditing to process the operation and facilitate the future auditing.

Step 2: Index update. Upon receiving the three types of update messages, the auditor calls three corresponding algorithms to update the I Table.

Step 3: Update confirmation. After the auditor updates the I Table, it conducts a confirmation auditing for the updated data and sends the result to

the owner. Then, the owner can choose to delete the local version of data according to the update confirmation auditing result.

6.3 Signature Generation Between TPA and Server

So we suggest employing a secure digital signature scheme to prevent the proof from being modified. Specifically, in Key Gen phase, the algorithm outputs additional two parameters (skS, pkS) as the cloud server’s secret/public key pair. In the auditing process, before sending the proof P = (TP, DP) to the auditor, the server uses its secret key skS to generate a signature σ of P and sends (TP, DP, σ) as the response to the challenge. Upon receiving the response, the auditor firstly verifies the signature σ . If it is valid, the auditor performs the Verify phase of the original auditing protocol; Otherwise, discards the response.

6.4 Fully Homomorphic Encryption Technique

Our ultimate goal is to construct a fully homomorphic encryption scheme ϵ . First, let us discuss what it means to be fully homomorphic.

At a high-level, the essence of fully homomorphic encryption is simple: given cipher texts that encrypt π_1, \dots, π_t , fully homomorphic encryption should allow anyone (not just the key-holder) to output a cipher text that encrypts $f(\pi_1, \dots, \pi_t)$ for any desired function f , as long as that function can be efficiently computed. No information about π_1, \dots, π_t or $f(\pi_1, \dots, \pi_t)$, or any intermediate plaintext values, should leak; the inputs, output and intermediate values are always encrypted.

Formally, there are different ways of defining what it means for the final cipher text to "encrypt" $f(\pi_1, \dots, \pi_t)$. The minimal requirement is correctness. A fully homomorphic encryption scheme should have an efficient algorithm Evaluate ϵ that, for any valid key pair (sk, pk) , any circuit C , and any cipher texts $\psi_i \leftarrow \text{Encrypt}_{\epsilon}(pk, \pi_i)$, outputs

$\psi \leftarrow \text{Encrypt}_{\epsilon}(pk, C, \psi_1, \dots, \psi_t)$ such that $\text{Decrypt}(sk, \psi) = C(\pi_1, \dots, \pi_t)$

Now that we have clarified our goal (fully homomorphic encryption), let us try to find a steppingstone. Suppose that, a priori, we have a scheme that is only guaranteed to be correct for some subset C_c of circuits - i.e.,

$\text{Decrypt}_{\epsilon}(sk, \text{Evaluate}_{\epsilon}(pk, C, \psi_1, \dots, \psi_t)) = C(\pi_1, \dots, \pi_t)$.

We also must maximize the evaluative capacity of the scheme, so that the scheme can evaluate its own (augmented) decryption circuit. While one can easily construct an additively homomorphic scheme from ordinary lattices, we need a scheme with both additive and multiplicative homomorphism's to evaluate general circuits. This consideration leads us to focus on ideal lattices.

7. CONCLUSION

To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We revisited the dynamic and privacy preserving auditing protocol for the cloud storage, proposed and demonstrated that an active adversary can modify the auditing proof to fool the auditor and the owner that the remote cloud files are pristine, while the files have been corrupted. In future work, we rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. New family of codes called Locally Repairable Codes (LRCs) that have marginally suboptimal storage. Our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the

distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s).

8. REFERENCES

- [1] G.-J. Ahn, H. Hu, Z. Hu, H. Wang, S.S. Yau, and Y. Zhu, 2011 "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," Proc. ACM Symp. Applied Computing, W.C. Chu, W.E. Wong, M.J. Palakal, and C.-C. Hung, eds., pp. 1550-1557.
- [2] G. Ateniese, L.V. Mancini, R.D. Pietro, and G. Tsudik, 2008 "Scalable and Efficient Provable Data Possession," IACR Cryptology ePrintArchive, vol. 2008, p. 114.
- [3] L.N. Bairavasundaram, G.R. Goodson, S. Pasupathy, and J. Schindler, 2007 "An Analysis of Latent Sector Errors in Disk Drives," Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems, L. Golubchik, M.H. Ammar, and M. Harchol-Balter, eds., pp. 289-300.
- [4] A. Birrell, M. Burrows, S. Elnikety, M. Isard and M. Lillibridge, 2003 "A Cooperative Internet Backup Scheme," Proc. USENIX Ann. Technical Conf., pp. 29-41.
- [5] C.C. Erway, A. Ku, "pc,u", C. Papamanthou, and R. Tamassia, 2009 "Dynamic Provable Data Possession," Proc. ACM Conf. Computer and Comm. Security, E. Al-Shaer, S. Jha, and A.D. Keromytis, eds., pp. 213-222.
- [6] G.R. Ganger, G.R. Goodson, M.K. Reiter and J.J. Wylie, 2004 "Efficient Byzantine-Tolerant Erasure-Coded Storage," Proc. Int'l Conf. Dependable Systems and Networks, pp. 135-144.
- [7] A. Juels and B.S. Kaliski Jr., "Pors: Proofs of Retrievability for Large Files, 2007" Proc. ACM Conf. Computer and Comm. Security, P. Ning, S.D.C. di Vimercati, and P.F. Syverson, eds., pp. 584-597.
- [8] V. Kher and Y. Kim, 2005 "Securing Distributed Storage: Challenges, Techniques, and Systems," Proc. ACM Workshop Storage Security and Survivability (StorageSS), V. Atluri, L. Brumbaugh, P. Samarati, W. Yurcik, and Y. Zhou, eds., pp. 9-25.
- [9] M.N. Krohn, J. Li, D. Mazieres, and D. Shasha, 2004 "Secure Untrusted Data Repository (SUNDR)," Proc. Sixth Conf. Symp. Operating Systems Design Implementation, pp. 121-136.
- [10] J. Li, W. Lou, K. Ren and C. Wang, 2010 "Toward Publicly Auditable Secure Cloud Data Storage Services," IEEE Network, vol. 24, no. 4, pp. 19-24, July/Aug.
- [11] W. Lou, K. Ren, C. Wang and Q. Wang, 2010 "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533.