Secure Public Auditing With Network Coding Based Storage in Cloud Architecture

[1]B.Lekha, [2]Prof. P. Kirubanantham
[1] P.G student, Thirumalai Engineering College, kilambi, kanchipuram, Tamil Nadu, India.
[2] Asst Prof. CSE, Thirumalai Engineering College, kilambi, kanchipuram, Tamil Nadu, India.

***Abstract* - Cloud computing setting in which probabilistic querying of outsourced data is a service provider. The data is to be accessed by only to the trusted users, not to the service distributer or anyone else. Outsourcing offers the data owner scalability. The need for privacy may be due to the data being sensitive or otherwise confidential. Still Security challenges and losing of information are the biggest problem when considering the adoption of cloud services. This triggered a several research activities, resulting in a quantity of proposals targeting the various cloud security threats and repair Storage node but doesn't provide efficient security mechanism and fault isolation process. Now this proposed system will overcome for all existing problems. New public auditing scheme for regenerating-code-based cloud storage, to solve the regeneration criticality problems of failed authenticators in the absence of data owners; to introduce a proxy, which is legally protected to regenerate the authenticators, into the traditional public auditing system model. Moreover, to design a novel public verifiable authenticator is generated by a couple of keys and using partial key for regeneration. Thus, our scheme can completely free that the data owners from online burden. In addition, our system using randomized coefficient which is encoded with a pseudorandom function to preserve data protection. Our scheme is to feasibly integrate into the regenerating-code-based cloud storage and it should be highly efficient.**

***Keywords* - public auditing, Cloud storage, regenerating codes, authenticator regeneration, proxy, provable secure.**

I. INTRODUCTION

Cloud computing is large group of network access to shared pool of configurable computing resources. It provides users and enterprises with various capabilities to store and process their data. Cloud storage is a necessary thing in various business activities and organizations for data accessing. Now-a-days it is gaining popularity because it provides a flexible on-demand data outsourcing service with various benefits: universal data access with location independence, effective storage management, and avoidance of capital expenditure on software, hardware, and storage maintenances etc., Due to larger network access the data storage may be easily theft by unauthorised users or corrupted by bandwidth problem, So the data owners lose their data, for that they affected their business and the main disadvantage of cloud is secure storage management.

The cloud service distributers may act dishonestly, tries to hide data corruption or loss and claiming that the files are still correctly stored in the cloud for reputation reasons. Thus it makes sense for user to implement an protocol to perform periodical verification of their data to ensure that the cloud indeed maintains their data effectively and correctly.

In this paper, we focus on the verification problem in regenerating-code-based cloud storage, in an traditional system uses particularly with the functional repair strategy, and the single-server Compact POR scheme to the regenerating-code-scenario in which designed and implemented a data integrity protection scheme for FMSR based cloud storage and the scheme is adapted to the thin-cloud setting However, above all schemes are designed for private audit, only the owner is allowed to verify the problem of integrity and repair the faulty servers. Considering the size of the outsourced data is high and the user's constrained resource capability, the tasks of auditing and reparation in the cloud can be difficult and exorbitant for the users. The overhead of using cloud storage should be minimized as much as possible such that a data owner does not need to perform many operations to their outsourced data. Especially the data owners may not want to go through the difficulty in verifying and reparation. The auditing schemes in CPOR and Data integrity protection imply the problem that users want to always stay online, which may delay and achieve long term storage in process to the users

To fully protect the data integrity and store the data owners' computation resources as well as online burden, we introduce 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 proxy separately on behalf of the data owner, in which the proxy is an semi trusted server. An alternative of directly adapting the existing public auditing scheme to the multi-server setting, we design an authenticator, which is more suitable for regenerating codes. Initially, we encrypt the coefficients to preserve data privacy against the auditor, this process is more lightweight. Several challenges can be solved in our system. This system having the following as aspects.

We design a homomorphic authenticator based on BLS signature, it can be generated by a couple of keys and regenerated using partial keys and which is verified publicly. The authenticators can be computed efficiently. our scheme is the first to allow secure public auditing with network coding based storage in cloud which defines regenerating-code-based cloud storage. The coefficients in users' data can be masked by a PRF (Pseudorandom Function) to avoid leakage of the original data. This method is more lightweight and does not arise any computational overhead to the cloud servers or TPA.

Our schemes completely free the data owners from online burden for the regeneration and authenticators at failure servers and it provides the legally protected to a proxy for the reparation. The storage overhead of cloud servers, the computational overhead of the data users and communication overhead during the audit phase can be completely reduced. Our scheme is completely secure under *random oracle model*. Further, we experimentally evaluate the performance of public auditing scheme.

II. PROBLEM STATEMENTS

System Model

We consider the auditing system model for network Coding based cloud storage as Fig.1, which involves four entities: *the data owner*, who owns data files to be stored in the cloud; *the cloud*, which are organised by the cloud service provider, provide storage service applications and computational resources; *the third party auditor* (TPA), who has capabilities to perform public audits on the coded data in the cloud, the TPA is trusted server and its audit result is unbiased for both cloud servers and data owners; and *a proxy agent*, who is semi-trusted server and acts as in support of the data owners to regenerate data blocks and authenticators on the failed servers during the repair procedure.

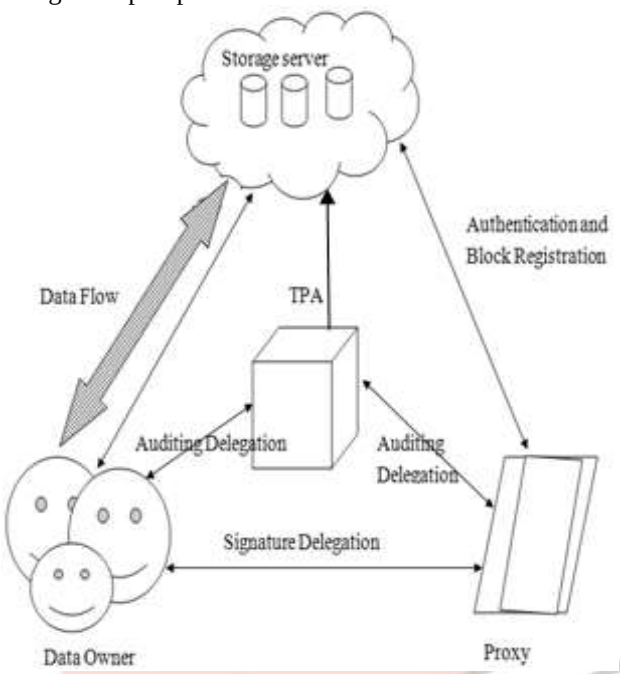

FIG.1 SYSTEM ARCHITECTURE

We know the user is restricted in storage resources and computational compared to other entities and may becomes off-line even after the data upload procedure.

The proxy server, who would always be online, is supposed to be more powerful than the data owner, but it is less than the cloud servers in terms of memory capacity and computation processes. To store resources as well as the staying and loading on online potentially brought by the periodic auditing and accidental repairing, the data owners resort to the Third Party Auditor for integrity verification and delegate the reparation to the proxy server.

Compared with the existing auditing system model, our system involves additional proxy server. This releases the data owners from online burden for data process and regeneration.

When estimate with the traditional public auditing system, our system architecture involves an additional proxy agent. It release the data owners from heavy online burden for authenticator regeneration and data, the organization supply a powerful workstation as the proxy server and provide proxy reparation service for the users' data.

Goals

The main goal of our System is to solve regenerating problem for cloud storage. In our system model has TPA to ensure that the third party auditor with secure and auditing efficiently. To verify the integrity of data effectively and keep the stored file available for cloud storage, our proposed public auditing scheme should have the following properties:

- *Public Auditability:* To allow Third Party Auditor 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 manage the owner's data intact.
- *Security:* To ensure both the auditor and proxy cannot derive users' data content from the auditing and reparation process.
- *Regeneration:* The authenticator can be correctly regenerate the repaired data block even in the absence of the data owner.
- *Error Location:* To ensure that the wrong server can be quickly indicated when data corruption is detected.

Procedures of Auditing Scheme

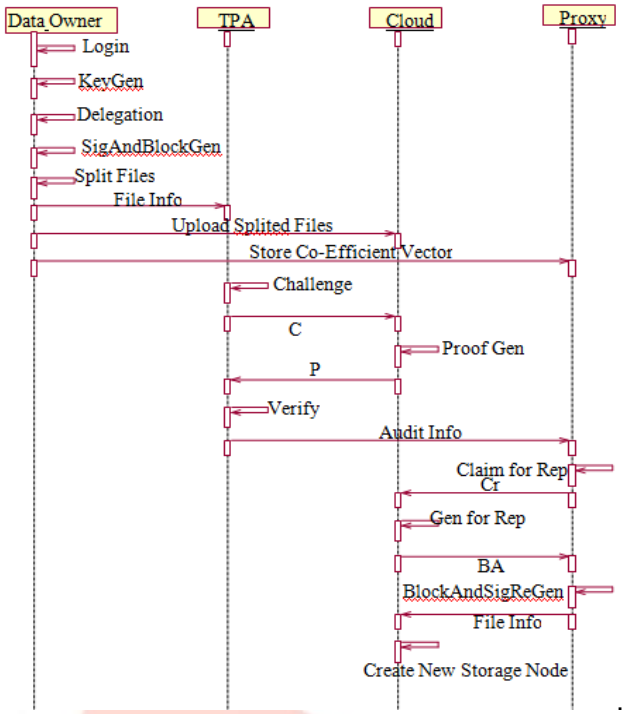

Fig 2. SEQUENCE DIAGRAM FOR AUDITING SCHEMES

Our public auditing scheme consists of three process: **Setup**, **Audit** and **Repair**. Each process having the procedure which contains specific polynomial-time algorithms as follows:

Setup: The data owner maintains the setup procedure to start the auditing scheme.
Audit: The cloud servers and Third Party Auditor(TPA) interact with one another to take a random sample on the data blocks and verify the data intactness in this step.
Repair: The proxy interacts with the cloud servers during repair procedure to repair the wrong server detected by the auditing process, even in the absence of data owner.

From the sequence diagram describes the procedures of our auditing scheme. In which that setup procedure initially create the secrete keys for the users data, the sigAndBlockGen algorithm takes the secrete keys and original file f and it will be split the files, then the partial secrete keys are taken to the proxy through a secure approach. the second procedure is an audit, this is perform by the third party auditor to audit the files, if any failure occurred in a auditing procedure due to some server problem. Then the repair procedure can be run by the proxy. It repair the wrong server even in the absence of data owner.

III. AUDITING SCHEME OVERVIEW

Although [3] introduced private remote data checking schemes for regenerating-code-based cloud storage, there are still some challenges for us to implementing a public auditable version.
Considering that a data owner maintains limited computation and little memory capacity, it is quite significant for us to diminish those overheads. Next, unlike cloud storage based on traditional erasure code or replication, a fixed file layout does not exist in the regenerating-code-based cloud storage. When the repair phase, it computes out new blocks, which are completely different from the faulty ones, with high probability.
Thus, a problem arises when trying to determine how to regenerate authenticators for the repaired blocks. A direct solution, which is adopted in [1], 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.
The following parts of this section shows our solution to the problems above. First, we construct a BLS-based [16] authenticator, which has of two parts for each segment of coded blocks. Utilizing its homomorphic property and the linearity relation among 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 [3] 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 operations in our auditing scheme work over the field $G\ F(\ p)$ of modulo p, where p is a large prime (at least 80 bits).

IV. PROPOSED SYSTEM MODULES

Setup Creation Module

The Setup creation module fully worked on data owner side and maintains this procedure to initialize the auditing scheme. The polynomial-time algorithm is using to initialize this process; it is run by the data owner.

KeyGen(1κ) → (pk, sk): This polynomial-time algorithm is run by the user initialize its public and secret parameters by taking a security parameter κ as input.

Degelation(sk) → (x): This algorithm represents the interaction between the user and proxy. The data owner delivers partial secret key *x* to the proxy through a secure approach.

SigAndBlockGen(sk, F) → (φ,ψ, 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 , an authenticator set and a file tag *t*.

File Upload Module

The file F is split into *m* blocks, and the original *m s*-dimensional vectors each original block wi is appended with the vector of length *m* containing a single '1' in the *i* th position and is otherwise zero.

Then, the augmented vectors are encoded into coded blocks. Specifically, they are linearly combined and generate coded blocks with randomly chosen coefficients vector.

Original encoded files upload into several storage node and coefficient vector has been stored into proxy server. The data server doesn't response that time will check vector and regenerate the missing data into new storage.

Public Auditing Module

The cloud servers and Third Party Auditor interact with one another to take a random sample on the blocks and check the data intactness in this procedure.

Challenge(Fin f o) → (C): This algorithm is performed by the TPA with the information of the file *Fin f o* as input and a *C* as output which defines challenge.

Proof Gen(C, φ,ψ)→(P)*:* This algorithm is run by each cloud server with input *C*, coded block set, authenticator set, then it outputs a proof .

P.Verify(P, pk, C)→ (0, 1): This algorithm is run by auditor immediately when a proof is received. Taking the proof *P*, public parameter *pk* and the corresponding *C* as input, its output is 1 if the verification passed and 0 otherwise.

Storage Node Re-Generate Module.

In the absence of the data owner, the proxy server interacts with the cloud servers during this procedure to repair the wrong server detected by the auditing process.

ClaimFor Rep(Fin f o) → (Cr): This algorithm is similar with the Challenge() algorithm in the Audit phase, but outputs a claim for repair *Cr* .

GenFor Rep(Cr, φ,ψ) → (BA): The cloud servers run this algorithm upon receiving the *Cr* and finally output the block and authenticators set *BA* with another two inputs

BlockAndSigReGen(Cr , BA) → (φ', ψ',⊥): The proxy implements this algorithm with the claim *Cr* and responses *BA* from each server as input, and outputs a new coded block set ψ' and authenticator set φ' if successful, outputting ⊥ if otherwise.

V. CONCLUSION

In this paper, we introduce an auditing scheme such that a public auditing for the regenerating-code-based cloud storage system, where the data owners are legally protected to delegate Third Party Auditor for their data validity verifying. To save the original data privacy against the Third Party Auditor, before an auditing process we randomize the coefficients. Considering that the data owner cannot always stay online in practise, we introduce a proxy server into the system model, We know that a Proxy is an semi trusted server and it is used to handle the reparation of the coded blocks and authenticators. To better suitable for the regenerating-code-scenario, we design our authenticator based on the (Boneh–Lynn–Shacham) BLS signature. Hence this authenticator providing the simultaneous encodes procedures and it is efficiently generated by the data owners. So our scheme is highly efficient, provable secure and integrated into a network coding based cloud storage system

REFERENCES

[1] A.Juels and B.Kaliski, Jr., "PORs: Proofs of retrievability for large files," in Proc. 14th ACM conf Comput. Commun. Secur., 2007 pp. 584-597.

[2] 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.

[3] 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.

[4] 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

[5] 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.

[6] C.Wang, Q. Wang, K. Ren, and W. Lou, "privacy- preserving public auditing for data storage security in cloud computing," in proc. IEEE INFORCOM, Mar. 2010, pp. 1-9.

[7] C.Wang, Q. Wang, N.Cao, K. Ren, and W. Lou, "Towards secure and dependable storage service in cloud computing," IEEE Trans. Service Comput., vol. 5, no.2 pp, 220-232, Apr./jun.2012.
[8] C.WANG, S.S.M. Chow, Q.Wang, K. Ren, and W. Lou, "privacy-preserving public auditing for secure cloud storage," IEEE Trans. Comput., vol.62 no.2, pp.362-375, Feb, 2013.
[9] T. HO et al., " A random linear network coding approach to multicast," IEEE Trans. Inf. Theory, vol. 52, no. 10, pp, 4413-4430, Oct, 2006.
[10] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, " Secure network coding over the integers," in Public Key Cryptography. Berlin, Germany: Springer-Verlag, 2010, pp. 142-160.
[11] Y. Deswarte, J.-J. Quisquater, and A. Saidane, " Remote integrity checking," in Integrity and Internal Control in Information Systems VI. Berlin, Germany: Springer-Verlag, 2004, pp. 1-11.
[12] Y. Dodis, S. Vadhan, and D.Wichs, "Proofs of retrivability via hardness amplification," in Theory of Cryptography. Berlin, Germany: Springer-Verlag, 2009, pp.109-127.
[13] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementations," in proc. ACM Workshop Cloud Comput. Secur., 2009, PP.43-54.
[14] S.G. Worku, C. Xu, J.Zhao, and X.He, "Secure and efficient privacy- preserving public auditing scheme for cloud storage," comput. Elect. Eng., vol. 40, no. 5, PP. 1703-1713, 2013.
[15] G. Ateniese, R. Di Pietro, L.V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession in proc. 4th Int. Conf. Secur. Privacy Commun. Netw., 2008, Art.ID 9.
[16] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," J. Cryptol., vol. 17, no. 4, pp.297-319, 2004.
[17] Q. Wang, C. Wang, J. Li, K. Ren, and W.Lou,, " Enabling public verifiability and data dynamics for storage security in cloud computing," in Computer Security. Berlin, Germany: Springer-Verlag, 2009, pp.355-370.