

Regenerating Code Based Cloud Storage using Privacy Preserving Public Auditing

Mrs. Swapnali Manish Nehete¹, Prof. Yogesh S. Patil², Prof. Dinesh D. Patil³

¹M.E. Student, Dept. of Computer Science and Engineering, SSGBCOET, Bhusawal, Maharashtra, India

²Assistant Professor, Computer Science and Engineering, SSGBCOET, Bhusawal, Maharashtra, India

³HOD, Computer Science and Engineering, SSGBCOET, Bhusawal, Maharashtra, India

ABSTRACT

The major scenario in Cloud Storage is, users can store their data and enjoy the on-demand high value applications and services from a common pool of configurable adding resources, without the problem of inherent data storage and conservation. To protect stored data in cloud storage against corruptions, adding fault tolerance to cloud storage together with data integrity checking and failure reparation becomes critical. However, the fact that users no longer have physical control over the authorized data makes the data reliability protection in Cloud Computing a tough task, especially for users with controlled computing resources. Also, users should be able to use the cloud loading as if it is local, without worrying about the need to verify its honesty. Thus, enabling public auditability for cloud storage is of critical position so that users can hand over to a third party auditor (TPA) to check the dependability of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new revelations towards user data privacy, and introduce no other online problem to user, we suggest a secure cloud storage system associate privacy-preserving public auditing. General security and performance analysis show that the proposed structures are provably secure and highly effective. Thus, our scheme can completely release data owners from online burden.

Keywords: Cloud storage, regenerating codes, Public auditing, Privacy Preserving, proxy

1. INTRODUCTION

Recently, regenerating codes have gained popularity due to their lower repair bandwidth while providing fault tolerance. 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. 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. The promise to deliver IT as a service is addressed a large range of consumers, from small and medium-sized enterprises (SMEs) and public administrations to end-users. According to industry analysts, the ICT sector is poised for strong growth of cloud services [11]. Users are creating an ever-growing quantity of personal data. IDC predicts the amount of information in the digital universe would fill a stack of iPad Air tablets reaching 2/3 of the way to the moon (253,704 Kilometer). By 2020, there will be 6.6 stacks. Today, the average household creates enough data to fill 65 iPhones (32 GB) per year. In 2020, this will grow to 318 iPhones. Today, if a byte of data were a gallon of water, in only 10 seconds there would be enough data to fill an average house. In 2020, it will only take 2 seconds.

2. EXISTING SYSTEM

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 significant work among these studies are the PDP (provable data possession) model and POR (proof of retrievability) model, which were originally proposed for the single-server scenario by Ateniese et al. and Juels and Kaliski, respectively. Considering that files are usually striped and redundantly stored across multi-servers or multiclouds, 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 and Chen separately and independently extended the single-server CPOR scheme to the re-generating code- scenario; Existing framework outlined and actualized an information trustworthiness assurance (DIP) plan for FMSR-based distributed storage and the plan is adjusted to the meager cloud setting. Be that as it may, the two are intended for private review, just the information owner is permitted to check the trustworthiness and repair the faulty data. Considering the expansive size of the outsourced information and the client's obliged asset ability, the undertakings of examining and reparation in the cloud can be impressive and costly for the clients. The overhead of utilizing distributed storage ought to be reduced however much as could reasonably be expected such that a client does not have to perform an excess of operations to their outsourced information. Specifically, clients might not have any desire to experience the unpredictability in checking and reparation.

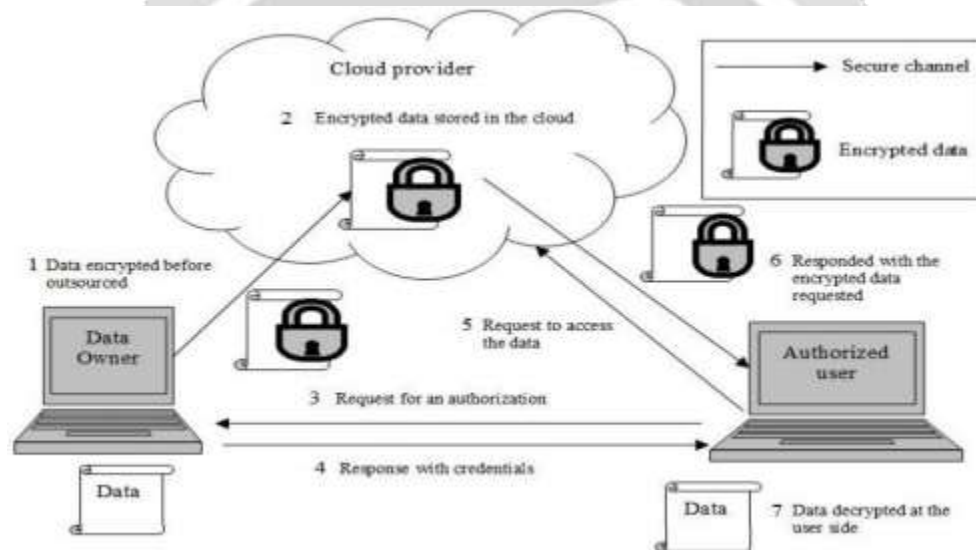


Figure 1. Existing System

3. DISADVANTAGES OF EXISTING SYSTEM

- To fully ensure the data integrity and save the users' computation resources as well as online burden.
- 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.

4. PROPOSED SYSTEM

We focus on the integrity verification problem in regenerating-code-based cloud storage, especially using **md5 sum verification** of the file. 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 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 and data blind method. We design a

novel homomorphic authenticator based on BLS signature, which can be generated by a couple of secret keys and verified publicly.

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. It generates signature using OAEP based key delegation which provides unique private and public key for each group registered in the cloud. So the users can access the document provided by its own group only. The users can view other group's document using private key of the other groups.

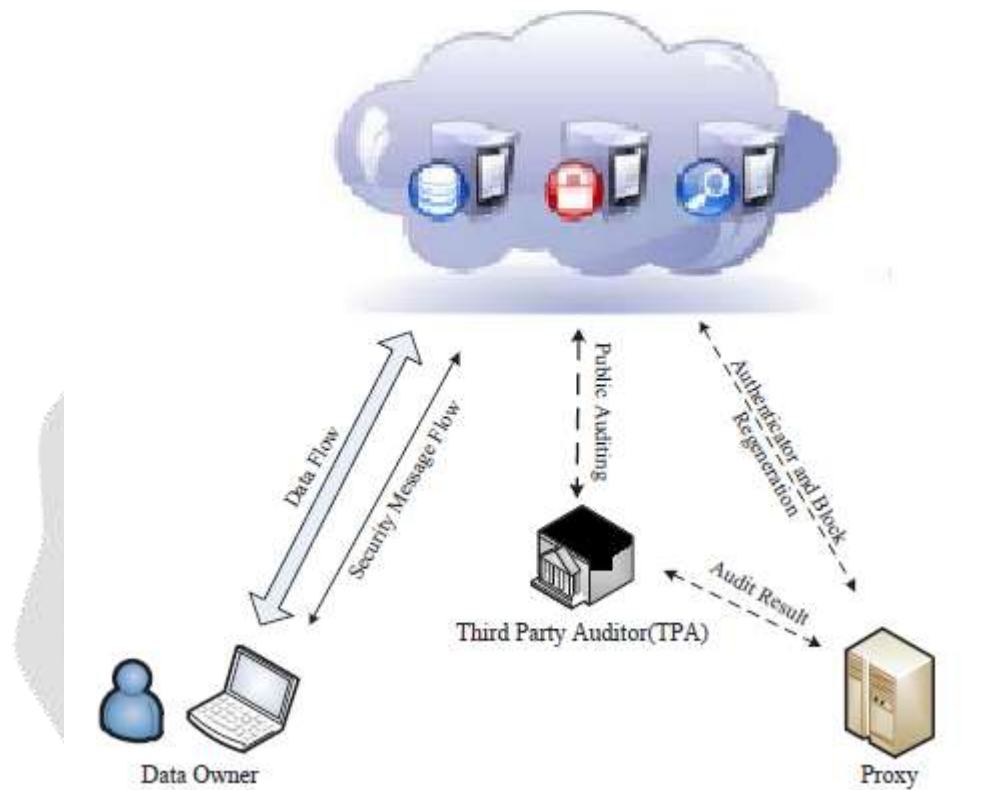



Figure 2. Proposed System

5. Typical working flow:

1. Data owner uploads the file to Cloud system and is stored in encrypted format.
2. Administrator Accepts/Rejects the file based on enforced policies.
3. Once the file is accepted, it is available for other users to use and download.
4. If some user requests a particular file he can select it from the list and request permission for it.
5. Data Owner decides if requester should access the file and sends a key to requester.
6. The key is only way with which the file can be accessed and decrypted.
7. This activity is logged in an audit report which is available to Admin.
8. TPA meanwhile can scan the cloud and verify files based on parameters
9. TPA can also validate files based on MD5 sum and suggest Admin for any corruption
10. Admin then restores the corrupted files from Semi-trusted Proxy and confirms back TPA
11. TPA revalidates corrupted blocks and gives go ahead.


6. Results



File Name **File Owner Name** **Size** **Decision**

PrimitiveParameters.bt	swapnali_rane14@gmail.com	31.0	Approved
File1.bt	myemployee02@gmail.com	17.0	Approved


Approve/Reject Page for Admin



File Audit


File Downloader	File Owner	File Name	Download Date
swapnali_rane14@gmail.com	swapnali_rane14@gmail.com	Demo-004.bt	2017-07-09

Download History for Admin page for Admin



File Name	File Owner Name	Size Proxy	Size Cloud	Action
PrimitiveParameters.bt	swapnali_rane14@gmail.com	31.0	31.0	Upload to Cloud
ed1.bt	swapnali_rane14@gmail.com	15.0	15.0	Upload to Cloud
File_Upload1.bt	myemployee02@gmail.com	24.0	24.0	Upload to Cloud
PrimitiveParameters.bt	swapnali_rane14@gmail.com	31.0	31.0	Upload to Cloud

Restore files from Proxy page for Admin



Regenerating code based cloud storage using
Privacy preserving Public Auditing

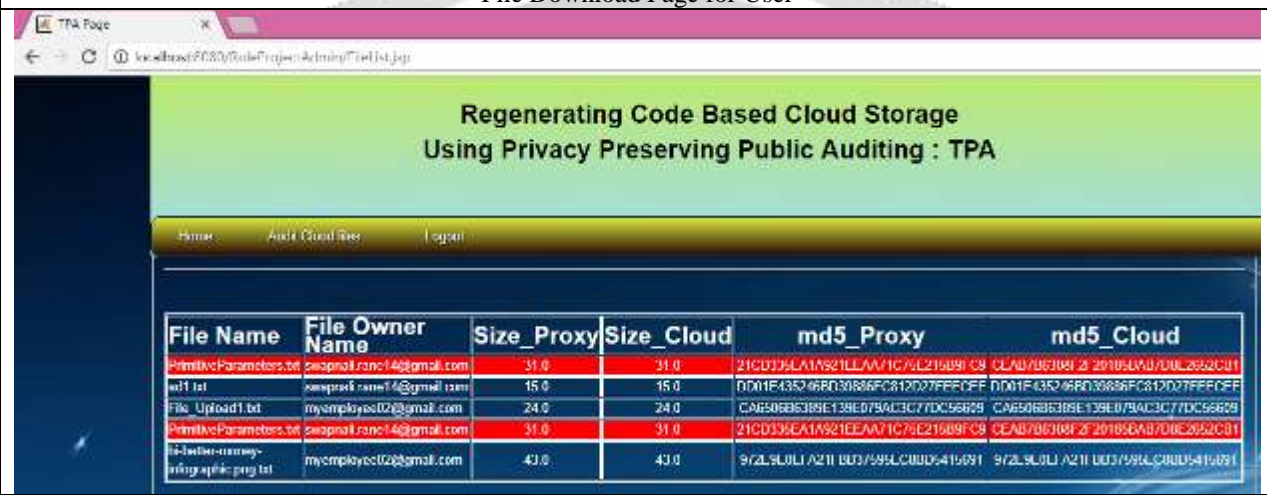
Home Upload File Download File File Uploaded Status Log Out

WELCOME MYEMPLOYEE02@GMAIL.COM

Download Files

File Name	File Owner Name	Operation
PrimitiveParameters.txt	swapnali.rane14@gmail.com	REQUEST
ad1.txt	swapnali.rane14@gmail.com	REQUEST
File1.txt	myemployee02@gmail.com	DOWNLOADED

File Download Page for User



TPA Page

Regenerating Code Based Cloud Storage
Using Privacy Preserving Public Auditing : TPA

Home Audit Cloud Files Logout

File Name	File Owner Name	Size_Proxy	Size_Cloud	md5_Proxy	md5_Cloud
PrimitiveParameters.txt	swapnali.rane14@gmail.com	31.0	31.0	21CD376A1A921EEAA71C7E21609FC9	CEAB70610BF2F20105B4B7D0E062C01
ad1.txt	swapnali.rane14@gmail.com	15.0	15.0	DD01E4352468D3086FC812D27FEEDFF	DD01E4352468D3086FC812D27FEEDFF
File_Upload1.txt	myemployee02@gmail.com	24.0	24.0	CAB506B63B9E139E079AC3C77DC56629	CAB506B63B9E139E079AC3C77DC56629
PrimitiveParameters.txt	swapnali.rane14@gmail.com	31.0	31.0	21CD376A1A921EEAA71C7E21609FC9	CEAB70610BF2F20105B4B7D0E062C01
File-Header-memo-graphic.png.txt	myemployee02@gmail.com	43.0	43.0	972LS0LI A211 B03/59ACBUD541691	972LS0LI A211 B03/59ACBUD541691

TPA page to validate files on Proxy and Cloud

7. Advantages of Proposed System

- 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.
- Universal data access with location independence
- Avoidance of capital expenditure on hardware, software, and personal maintenances, etc.,

8. CONCLUSION

It's 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. 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 to admin to restore any corruption of data. In addition to this we have added intelligent Captcha which would provide additional security against bloatware and brute force attack. Extensive analysis shows that our scheme is provable 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.

9. REFERENCES

- [1] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data regeneration scheme for cloud storage," in Technical Report, 2013.
- [2] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from regenerate files in a serverless distributed file system," in ICDCS, 2002, pp. 617– 624.
- [3] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-regeneration," in Proc. of USENIX LISA, 2010.
- [4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Serveraided encryption for deregenerated storage," in USENIX Security Symposium, 2013.
- [5] G. R. Blakley and C. Meadows, "Security of ramp schemes," in Advances in Cryptology: Proceedings of CRYPTO '84, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds. Springer-Verlag Berlin/Heidelberg, 1985, vol. 196, pp. 242–268.
- [6] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure regeneration with efficient and reliable convergent key management," in IEEE Transactions on Parallel and Distributed Systems, 2014, pp. vol. 25(6), pp. 1615–1625.
- [7] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in ACM Conference on Computer and Communications Security, Y. Chen, G. Danezis, and V. Shmatikov, Eds. ACM, 2011, pp. 491–500.
- [8] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, "R-admad: High reliability provision for large-scale de-regeneration archival storage systems," in Proceedings of the 23rd international conference on Supercomputing, pp. 370–379.
- [9] M. Li, C. Qin, P. P. C. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," in The 6 th USENIX Workshop on Hot Topics in Storage and File Systems, 2014.
- [10] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-solomon Codes for fault-tolerant network storage applications," in NCA- 06: 5 th IEEE International Symposium on Network Computing Applications, Cambridge, MA, July 2006.
- [11] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Regeneration in cloud.
- [12] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applications - Version 1.2," University of Tennessee, Tech. Rep. CS-08-627, August 2008.
- [13] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," Journal of the ACM, vol. 36, no. 2, pp. 335–348, Apr. 1989.
- [14] A. Shamir, "How to share a secret," Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.