International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume I, Issue VI, OCT - 2014

KEY AGGREGATE DATA STORAGE AND DATA INTEGRITY PROTECTION IN CLOUD

V.Rajkumar
Department of Computer Science and Engineering
Nandha College of Technology
Erode, India
rajkumarit30@gmail.com

D.Mohanapriya
Department of Computer Science and Engineering
Nandha College of Technology
Erode, India
spriyasasmi@gmail.com

Abstract - To protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage, along with efficient data integrity checking and recovery procedures, becomes critical. We design and implement a practical data integrity protection (DIP) scheme for a specific regenerating code, while preserving its intrinsic properties of fault tolerance and repair-traffic saves. Our DIP scheme is designed under a mobile Byzantine adversarial model, and enables a client to feasibly verify the integrity of random subsets of outsourced data against general or malicious corruptions. It works under the simple assumption of thin-cloud storage and allows different parameters to be fine-tuned for a performance-security trade-off. We implement and evaluate the overhead of our DIP scheme in a real cloud storage test bed under different parameter choices. We further analyze the security strengths of our DIP scheme via mathematical models. We demonstrate that remote integrity checking can be feasibly integrated into regenerating codes in practical deployment.

I. INTRODUCTION

Cloud storage offers an on-demand data outsourcing service model, and is gaining popularity due to its elasticity and low maintenance cost. However, security concerns arise when data storage is outsourced to third-party cloud storage providers. It is desirable to enable cloud clients to verify the integrity of their outsourced data, in case their data have been accidentally corrupted or maliciously compromised by insider/outsider attacks. One major use of cloud storage is long-term archival, which represents a workload that is written once and rarely read. While the stored data are rarely read, it remains necessary to ensure its integrity for disaster recovery or compliance with legal requirements Since it is typical to have a huge amount of archived data, whole-file checking becomes prohibitive. Proof of irretrievability (POR) and proof of data possession (PDP) have thus been proposed to verify the integrity of a large file by spot-checking only a fraction of the file via various crypto-graphic primitives.

Cloud Data authentication: Data authentication ensures the group member that the data was accessed by a specified owner and the data was not altered en route. To provide these two functions, Dynamic Group key protocol relies on one trusted entity, KGC (Key Generation Center), to choose the key, which is then transported to each member involved. Each user is required to register at KGC for subscribing the key distribution service. The KGC keeps tracking all registered users and removing any unsubscribed users through revocation. The dynamic nature allows the members to leave and join the group and instead of performing individual rekeying operations due to key sharing overload and complexity, so the system uses interval based Interval Based algorithm for re-keying which not only reduce the key generation and sharing complexity but also improves the owner data sharing efficiency and security.

It repairs traffic saving over traditional erasure codes. A related approach is HAIL , which applies integrity protection for erasure codes. It constructs protection data on a per-file basis and distributes the protection data across different servers. To repair any lost data during a server failure, one needs to access the whole file, and this violates the design of regenerating codes. Thus, we need a different design of integrity protection tailored for regenerating codes.
In this paper, we design and implement a practical data integrity protection (DIP) scheme

for regenerating-coding-based cloud storage. We augment the implementation of functional minimum-storage regenerating (FMSR) codes [15] and construct FMSR-DIP codes, which allow clients to remotely verify the integrity of random subsets of long-term archival data under a multi server setting. FMSR-DIP codes preserve fault tolerance and repair traffic saving as in FMSR codes [15]. Also, we assume only a thin-cloud interface [27], meaning that servers only need to support standard read/ write functionalities. This adds to the portability of FMSR-DIP codes and allows simple deployment in general types of storage services. By combining integrity checking and efficient recovery, FMSR-DIP codes provide a low-cost solution for maintaining data availability in cloud storage.

In summary, we make the following contributions:

- We design FMSR-DIP codes, which enable integrity protection, fault tolerance, and efficient recovery for cloud storage.
- .We export several tunable parameters from FMSR-DIP codes, such that clients can make a trade-off between performance and security.
- .We conduct mathematical analysis on the security of FMSR-DIP codes for different parameter choices.
- .We implement FMSR-DIP codes, and evaluate their overhead over the existing FMSR codes through extensive tested experiments in a cloud-storage environment. We evaluate the running times of different basic operations, including Upload, Check, Download, and Repair, for different parameter choices.

le auditing tasks simultaneously. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

The FMSR codes and construct FMSR-DIP codes, which allow clients to remotely verify the integrity of random subsets of long-term archival data under a multiserver setting. FMSR-DIP codes preserve fault tolerance and repair traffic saving as in FMSR codes. Also, we assume only a thin-cloud interface meaning that servers only need to support standard read/ write functionalities. This adds to the portability of FMSRDIP codes and allows simple deployment in general types of storage services.

III. METHODS

The remainder of the paper proceeds as follows: Section 2 reviews related work. Section 3 provides necessary preliminaries for our DIP design. Section 4 presents the design of FMSR-DIP codes. Section 5 presents the implementation details and discusses how parameters can be adjusted for different performance needs. Section 6 presents security analysis for FMSR-DIP codes. Section 7 reports evaluation results of FMSR-DIP codes in a cloud-storage tested. Finally, Section 8 concludes the paper. We also refer readers to our digital supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/ TPDS.2013.164, for additional discussion of this work.

II. RELATED WORK

Providing simultaneous public audit ability and data dynamics for remote data integrity check in Cloud Computing. The existing method construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic functional minimum-storage regenerating (FMSR) construction for block tag authentication. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multiuser setting, where TPA can perform multip

- USER INTERFACE DESIGN
- CLOUD STORAGE
- CHECK OPERATION
- REPAIR OPERATION
- THIRD PARTY AUDITOR
- CLOUD CLIENT
- GROUP MEMBER MODULE

IV. USER INTERFACE DESIGN

The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. Graphic design may be utilized to support its usability. The

design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs. Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge.

V. CLOUD STORAGE

Cloud Storage is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. Hosting companies operate large data centers; and people who require their data to be hosted buy or lease storage capacity from them and use it for their storage needs. The data center operators, in the background, virtualized the resources according to the requirements of the customer and expose them as virtual servers, which the customers can themselves manage. Physically, the resource may span across multiple servers.

VI. REPAIR OPERATION

If some server fails (e.g., when losing all data or having too much corrupted data that cannot be recovered), we trigger the Repair operation via NCCloud as follows:
Step 1: Check the metadata file. Refer to Step 1 of Check.
Step 2: Download and decode the needed chunks. This is similar to Step 2 of Download, as long as there are at most n _ k failed servers. In particular, if there is only one failed server, then instead of trying to download kðn _ kÞ chunks from any k servers, we download one chunk from all remaining n _ 1 servers as in FMSR codes.
Step 3: Encode, update metadata, and upload. NCCloud generates n _ k chunks to store at the new server. Each chunk is encoded with FMSR-DIP codes again (Step 3 of Upload) and uploaded to the new server. Finally, the metadata is updated, encrypted, and replicated to all servers (Step 4 of Upload).

VII. THIRD PARTY AUDITOR

TPA in possession of the public key can act as a verifier. We assume that TPA is unbiased while the server is untrusted. For application purposes, the clients may interact with the cloud servers via CSP to access or retrieve their pre-stored data. More importantly, in practical scenarios, the client may frequently perform block-level operations on the data files. The most general forms of these operations we consider in this paper are modification, insertion, and deletion.Public auditability for storage correctness assurance: to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.
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.Blockless verification: no challenged file blocks should be retrieved by the verifier (*e.g.*, TPA) during verification process for efficiency concern.

VIII. CLOUD CLIENT

A cloud client consists of computer hardware and/or computer software that relies on cloud computing for application delivery, or that is specifically designed for delivery of cloud services and that, in either case, is essentially useless without it. Examples include some computers, phones and other devices, operating systems and browsers.

IX. GROUP MEMBER MODULE

Group members are a set of registered users that will store their private data into the cloud server and share them with others in the group. The group membership is dynamically changed, due to the staff resignation and new employee participation in the company. Data owners generate data and upload them to the

cloud for sharing. Data users are able to access data uploaded by data owners. So after receiving public key from the manager it get data access in the cloud system then act as multi-owner. Then create private key to access their data in the cloud which is transferred to the authorized members in the group. Thus authorized members are capable to update or delete the data with that key under multi-owner.

File Access

Any group member can store and share data files with others in the group by the cloud. User revocation can be achieved without involving the remaining users. That is, the remaining users do not need to update their private keys or re encryption operations. New granted users can learn all the content data files stored before his participation without contacting with the data owner.

X. PROPOSED METHOD

The FMSR codes and construct FMSR-DIP codes, which allow clients to remotely verify the integrity of random subsets of long-term archival data under a multiserver setting. FMSR-DIP codes preserve fault tolerance and repair traffic saving as in FMSR codes. Also, we assume only a thin-cloud interface meaning that servers only need to support standard read/ write functionalities. This adds to the portability of FMSRDIP codes and allows simple deployment in general types of storage services.

We solve this problem by introducing a special type of public-key encryption which we call key-aggregate cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of cipher text called class. That means the cipher texts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of cipher text classes.

The sizes of ciphertext, public-key, master-secret key, and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of ciphertext classes, but only a small part of it is needed each time and it can be fetched on demand from large (but nonconfidential) cloud storage. Previous results may achieve a similar property featuring a constant-size decryption key, but the classes need to conform to some predefined hierarchical relationship. Our work is flexible in the sense that this constraint is eliminated, that is, no special relation is required between the classes.

XI. CONCLUSION

Given the popularity of outsourcing archival storage to the cloud, it is desirable to enable clients to verify the integrity of their data in the cloud. We design and implement a DIP scheme for the FMSR codes under a multiserver setting. We construct FMSR-DIP codes, which preserve the fault tolerance and repair traffic saving properties of FMSR codes. To understand the practicality of FMSRDIP codes, we analyze the security strength via mathematical modeling and evaluate the running time overhead via testbed experiments. We show how FMSR-DIP codes trade between performance and security under different parameter set-tings. The source code of the implementation of our FMSR-DIP codes is available at: http://ansrlab.cse. cuhk.edu.hk/ software/fmsrdip.

REFERENCE

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, “RACS: A Case for Cloud Storage Diversity,” Proc. First ACM Symp. Cloud Computing (SoCC ’10), 2010.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. [26] “TechCrunch,” Online Backup Company Carbonite Loses Custom Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A View of Cloud Computing,” Comm. ACM, vol. 53, no. 4, pp 50-58, 2010.

[3] Data, Blames and Sues Suppliers. K. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), 2009.
[4] K. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Comput-ing Security (CCSW '09), 2009.
[5] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-Based Distributed Storage Sys-tems," Proc. ACM Workshop Cloud Computing Security (CCSW '10),2010.
[6] H.C.H. Chen and P.P.C. Lee, "Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage," Proc. IEEE 31st Symp. Reliable Distributed Systems (SRDS '12), 2012.
[7] L. Chen, "NIST Special Publication 800-108," Recommendation for Key Derivation Using Pseudorandom Functions (Revised), http:// csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf, Oct. 2009.
[8] R. Curtmola, O. Khan, and R. Burns, "Robust Remote Data Checking," Proc. ACM Fourth Int'l Workshop Storage Security and Survivability (StorageSS '08), 2008.
[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. IEEE 28th Int'l Conf. Distributed Computing Systems (ICDCS '08), 2008.
[10] Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Sys-tems," IEEE Trans. Information Theory, vol. 56, no. 9, 4539-4551, Sept. 2010.
[11] D. Ford, F. Labelle, F.I. Popovici, M. Stokel, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," Proc. Ninth USENIX Symp. Operating Systems Design and Implementation (OSDI '10), Oct. 2010.
[12] O. Goldreich, Foundations of Cryptography: Basic Tools. Cambridge Univ. Press, 2001.
[13] O. Goldreich, Foundations of Cryptography: Basic Applications.
[14] Cambridge Univ. Press, 2004.
[15] Y. Hu, H. Chen, P. Lee, and Y. Tang, "NCCloud: Applying Network Coding for the Storage Repair in a Cloud-of-Clouds," Proc. 10th USENIX Conf. File and Storage Technologies (FAST '12).
[16] Juels and B. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), 2007.
[17] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," Proc. 30th Ann. Conf. Advances in Cryptology (CRYPTO '10), 2010.
[18] E. Naone, "Are We Safeguarding Social Data?" http:// www.technologyreview.com/blog/editors/22924/, Feb. 2009.
[19] J.S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-Like Systems," Software - Practice & Experience, vol. 27, no. 9, pp. 995-1012, Sept. 1997.
[20] M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," J. ACM, vol. 36, no. 2, pp. 335-348, Apr. 1989.
[21] Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," J. Soc. Industrial and Applied Math., vol. 8, no. 2, pp. 300-304, 1960.
[22] B. Schroeder, S. Damouras, and P. Gill, "Understanding Latent Sector Errors and How to Protect against Them," Proc. USENIX Conf. File and Storage Technologies (FAST '10), Feb. 2010.