

IN CLOUD STORAGE ENVIRONMENT DATA STORAGE AMONG DROPBOX AND BITCASA TO INCREASE DATA PERSISTENT

¹E.Fathima, ² Dr.C.Rajabhushanam, ³K.Sivaraman

^{1,3}Assistant Professor, Dept. Of CSE, BIST,BIHER,Bharath University

²Professor, Dept. Of CSE, BIST,BIHER,Bharath University

¹fathima.cse@bharathuniv.ac.in

Abstract: In cloud storage environment data is stored among storage boxes such as Drop box and Bitcasa to increase the data persistent. Currently, with prevalence of mobile computing the user can incorporate to edit the newest version of documents and files on their smart mobile devices. To support unlimited storage the cloud storage provider utilizing data deduplication technique to reduce the data to be stored and therefore reduce the storage expense the use of data deduplication also helps significantly reduce the need for bandwidth and therefore improve the user experience. Nevertheless, in spite of the above benefits, data deduplication has its inherent security weaknesses. Among them, the most severe is that the adversary may have an unauthorized file downloading via the file hash only. In this article we first review the previous solutions and identify their performance weaknesses. Then we propose an alternative design that achieves cloud server efficiency and especially mobile device efficiency.

1. Introduction

Cloud computing is envisaged to play key roles of motivation in the design of the future smart grid. Cloud computing is an emerging technology advocated for enabling convenient, on-demand network access to shared computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Using cloud infrastructure, a customer can gain access to their applications anytime, from anywhere, through a connected device to the network. cloud computing is not limited to localized isolated systems[1-3]. In case of large computing or storage resources, the mobile nodes request services from the cloud servers. A mobile health-cloud is an integration of health-cloud into the mobile environment. In a mobile health-cloud environment, the data processing and storage applications are moved from the mobile body-sensor

nodes to the servers situated in the health-cloud. The integration of health-cloud into the mobile environment improves data computing and storage facility, which leads to the prolongation of lifetime of batteries in the mobile body sensor nodes. The cloud service providers have to spend a lot of human and financial resources to store and manage these data. Even worse, according to literature[4], more than 70% data in cloud storage are duplicated data. To relieve the storage stress of cloud service providers and to make data management scalable, deduplication becomes a basic requirement in cloud storage systems [5-6].

2. Security Weakness

However, client side deduplication incurs its own security weaknesses. First, the privacy of the file existence in the cloud may be compromised because the adversary may try to upload the candidate files to see whether the deduplication takes place. If the deduplication takes place, this will be an indicator of the file's existence. Otherwise, the adversary may infer the file's nonexistence. The situation becomes even worse when we consider the low-entropy files because the adversary may exhaustively create different files and upload the hashes to check the file's existence. For example, a curious colleague may query his/her manager's salary by uploading different salary sheets because the sheets are of a similar form, restricting the number of file contents to be tested[7-9]. The deduplicated cloud storage may be abused as a covert channel. By agreeing upon a common file beforehand, two users may exchange the bit information via the file status. In particular, if bit 1 needs to be sent, then a user uploads the file. Another user detects the deduplication, interpreting the deduplication as a bit 1. If bit 0 needs to be sent, the file is removed from the cloud and the deduplication does not take place at another user. Despite its very low bandwidth, such a covert channel may be used as a botnet command and control (C&C) channel

for the botmaster to dispatch the botnet controls in a stealthy way that can also penetrate the firewall and escape the deep packet inspection. Third, the deduplicated cloud storage may be abused as a can gain the file's ownership and then download it by uploading the hash [10-12].

3. Proof of Ownership

An even more severe and direct security threat from using deduplicated cloud storage is that the adversary may gain the ownership of files by only eavesdropping on file hashes. A closer look at client side deduplication can find that anyone in possession of the file hash can gain ownership of the file by uploading the file hash. More specifically, the cloud considers receiving a store request for a file already in the storage, avoids the redundant file transmission, and then adds the user as an additional owner of the file. Such a situation is apparently undesirable because in theory the adversary cannot infer the file content via the hash. However, in this case, once the adversary knows the hash, it is able to download the entire file content. On the other hand, in practice, the user considers the hash unharmed and in some cases publishes the hashes as timestamps. However, the publicly available hashes can be abused to gain the file. This security weakness comes from using the static and short piece of information (hash) as a way of claiming file ownership. the cloud and user such that the user can prove to the cloud that it is indeed in possession of the file[13-14].

4. Design Objective

There are many metrics to evaluate the efficiency of a POW scheme such as the bandwidth requirement, I/O overhead at both user and server sides, and the computation overhead at both sides. While the first and third obviously concern the performance, the second is less known in the POW design. More specifically, cloud storage usually has a storage hierarchy. The execution of a POW scheme might require the user and cloud to access the file stored in the disk multiple times. The server might also need to keep the verification object in either the memory or the disk to verify the user's claim. The above all might result in a huge amount of I/O delay because of the access time gap between the memory and disk. In this article we focus only on the abuse of a file hash to gain the ownership of the file and aim to design a POW scheme with minimum performance overhead[15-17].

5. Prior Art

A series of modifications to the general universal hash function used in the second construction, trying to further improve the computation efficiency the third construction does not have provable security guarantee, and can only be secure for high-entropy files. Though the POW schemes have low storage overhead and incur no I/O delay at the cloud, the user needs to compute the Merkle hash tree, causing an unavoidable computation burden and I/O delay at the user. Aiming at improving the user side efficiency, also propose three POW schemes: s-POW, s-POW1, and s-POW2. The rationale behind the three schemes is even simpler. Once the user claims ownership of the file f , it needs to be able to correctly answer every bit of f . s-POW implements the above simple idea. After receiving and storing the first copy of the file f , the cloud randomly chooses certain positions and keeps a number of the corresponding bits of f in the memory. Upon receiving subsequent requests, the cloud issues the challenge about the bits of f . With the observations that the collision-free hash function used for checking whether the deduplication takes place can be replaced with either the random bits of the file or even the file size only design s-POW1 and s-POW2, respectively, by trading off the deduplication accuracy and computation efficiency.

Despite the efficiency at the user side, the server has substantial I/O burden, contrary to the POW schemes in Specifically, since the positions of the file to be queried need to be random, once a file is requested multiple times, all bits of f in the memory will be used up and therefore the cloud is forced to fetch more bits from the disk. Note that because of the memory space constraint, we can only store some sampled bits.

The Proposed POW Scheme

The POW schemes in performance very well on the server side since only a small size index (tree root) needs to be stored in the main memory. However, the proof of ownership is achieved by the user sending an authentication path of size $O(\log |f|)$ to the cloud, resulting in more communication overhead and computation load on the cloud. The I/O overhead of the user side is also increased, compared to the POW schemes in , because the user needs to retrieve the entire file. On the other extreme, although the s-POW schemes in have great computation and I/O efficiency in the user side, its I/O burden on the cloud is significantly increased since the cloud is required to retrieve random bits from the secondary storage.

In this article we propose an alternative design that strikes a balance between server side efficiency and user side efficiency. Before introducing the s scheme's details, we present two observations. First, the POW schemes in are I/O efficient at the server side because the Merkle tree root can be thought of as a compact summary of the file. Therefore, there is no need for the cloud to access the disk

to retrieve the file. Second, the user side is computationally efficient in three s-POW schemes because the user is simply required only to answer a few bits of the file. With the above two observations, our design strategy is to have a probabilistic data structure for the compact summary of the file, in contrast to the deterministic data structure, Merkle hash tree, in the POW schemes. The query challenge is also modified as random blocks, in contrast to the random bits in s-POW schemes. An overview of the proposed POW scheme goes as follows[18-20].

During the cloud storage initialization, the cloud constructs an empty Bloom filter. Note that Bloom filter B is kept in the memory. Upon receiving the first copy of the file f , the cloud partitions f into blocks of fixed length l . Here we simply assume the Upon receiving the subsequent request, the cloud checks whether $h(f)$ is already in the memory. If not, the cloud simply ignores the request. Bloom filter, to accommodate the deletion operation without compromising the false positive probability. Note that the counting Bloom filter is a natural extension of the ordinary Bloom filter, where each bit is replaced with a counter. The counting Bloom filter supports insertion and deletion operations since each counter can be incremented and decremented. For the latter case, we use the dynamic Bloom filter to keep the use of memory to a minimum and keep the false positive probability low. A remarkable feature of the dynamic Bloom filter is that in the case of full capacity, the length of the dynamic Bloom filter can be extended to accommodate more elements. the dynamic Bloom filter is in essence a variant of the counting Bloom filter. Thus, it is sufficient to handle the above issues by replacing the ordinary Bloom filter with the dynamic Bloom filter[21]. We omit a detailed description because of its similarity to the POW scheme with the ordinary Bloom filter. is larger than the tree root in , our scheme maintains the advantage that the server has no need to access the disk after storing f because the block information about f has all been kept in B , which is similar to the POW schemes the other hand, the user needs to access different blocks of the file and calculate the hashes, resulting in more computation overhead compared to the one in s-POW schemes. it is much more efficient than the POW schemes in that need to access every bit of the file and perform a lot of hash computation. The performance comparison can be found in Some simple analysis is sufficient to demonstrate the performance improvement. Assume that the cloud issues a challenge about l random bits in , l random random leaf positions in and l random blocks in our POW scheme. It is sufficient for the cloud to verify the user's claimed ownership by using B in the

memory only. Thus, no disk I/O is needed. Assuming hash functions used in B , since the cloud needs to check the membership of l random blocks in B , lk hash computations are necessary[22].

6. Conclusion

Aiming to minimizing both the server side latency and the user side latency, we propose an alternative POW design on the problem of unauthorized file downloading in deduplicated cloud storage. In our design, the use of probabilistic data structure, the Bloom filter, primarily contributes to the overhead reduction. Since the Bloom filter has been used widely in various applications and is easy to be implemented, our proposed POW scheme is considered realistic and can be deployed in real-world cloud storage services. Despite the use of the Bloom filter in reducing the I/O needs, the size of the Bloom filter may grow with the number of files stored in the cloud. The Bloom filter may also be of a huge size so that it needs to be partitioned and part of it needs to be stored in the disk. Thus, one possible future research focus is to develop a more succinct data structure or devise a new index mechanism such that the index (the Bloom filter in this article) can be fit into the memory even in the case of a huge number of files in the cloud.

References

- [1] Jiawei Yuan And Shucheng Yu, "Secure and Constant Cost Public Cloud Storage Auditing with Deduplication"2015 IEEE Conference on Communications and Network Security(CNS).
- [2] Snigdha Das, Sudip Misra, Manas Khatua and Joel J. P. C. Rodrigues, "Mapping of Sensor Nodes with Servers in a Mobile Health-Cloud Environment"2015 IEEE International conference on e-Health Networking, Applications and Services.
- [3] Fatema Rashid, Ali Miri, and Isaac Woungang, "Proof of Storage for Video Deduplication in the Cloud" 2014IEEE International Congress on Big Data.
- [4] Samareesh Bera, Sudip Misra, Senior Member, IEEE, and Joel J. P. C. Rodrigues, Senior Member, IEEE, "Cloud Computing Applications for Smart Grid: A Survey", 2014IEEE Trans.Parallel Distrib.Syst., to be published.
- [5] Ruiying Du, Lan Deng, Jing Chen, Kun He and Minghui Zheng, "Proofs of Ownership and Retrievalability in Cloud Storage" 2013IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications.
- [6] Sudip Misra, Senior Member, IEEE, Snigdha Das, Manas Khatua, Student Member, IEEE, and Mohammad S. Obaidat, Fellow, IEEE, "QoS -Guaranteed Bandwidth

Shifting and Redistribution in Mobile Cloud Environment". IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 2, APRIL-JUNE 2013.

[7] Udayakumar R., Kaliyamurthie K.P., Khanaa, Thooyamani K.P., Data mining a boon: Predictive system for university topper women in academia, World Applied Sciences Journal, v-29, i-14, pp-86-90, 2014.

[8] Kaliyamurthie K.P., Parameswari D., Udayakumar R., QOS aware privacy preserving location monitoring in wireless sensor network, Indian Journal of Science and Technology, v-6, i-SUPPL5, pp-4648-4652, 2013.

[9] Brintha Rajakumari S., Nalini C., An efficient cost model for data storage with horizontal layout in the cloud, Indian Journal of Science and Technology, v-7, i-, pp-45-46, 2014.

[10] Brintha Rajakumari S., Nalini C., An efficient data mining dataset preparation using aggregation in relational database, Indian Journal of Science and Technology, v-7, i-, pp-44-46, 2014.

[11] Khanna V., Mohanta K., Saravanan T., Recovery of link quality degradation in wireless mesh networks, Indian Journal of Science and Technology, v-6, i-SUPPL.6, pp-4837-4843, 2013.

[12] Khanaa V., Thooyamani K.P., Udayakumar R., A secure and efficient authentication system for distributed wireless sensor network, World Applied Sciences Journal, v-29, i-14, pp-304-308, 2014.

[13] Udayakumar R., Khanaa V., Saravanan T., Saritha G., Retinal image analysis using curvelet transform and multistructure elements morphology by reconstruction, Middle - East Journal of Scientific Research, v-16, i-12, pp-1781-1785, 2013.

[14] Khanaa V., Mohanta K., Saravanan. T., Performance analysis of FTTH using GEAPON in direct and external modulation, Indian Journal of Science and Technology, v-6, i-SUPPL.6, pp-4848-4852, 2013.

[15] Kaliyamurthie K.P., Udayakumar R., Parameswari D., Mugunthan S.N., Highly secured online voting system over network, Indian Journal of Science and Technology, v-6, i-SUPPL.6, pp-4831-4836, 2013.

[16] Thooyamani K.P., Khanaa V., Udayakumar R., Efficiently measuring denial of service attacks using appropriate metrics, Middle - East Journal of Scientific Research, v-20, i-12, pp-2464-2470, 2014.

[17] R.Kalaiprasath, R.Elankavi, Dr.R.Udayakumar, Cloud Information Accountability (Cia) Framework Ensuring Accountability Of Data In Cloud And Security In End To End Process In Cloud Terminology, International Journal Of Civil Engineering And Technology (Ijciety) Volume 8, Issue 4, Pp. 376–385, April 2017.

[18] R.Elankavi, R.Kalaiprasath, Dr.R.Udayakumar, A fast clustering algorithm for high-dimensional data,

International Journal Of Civil Engineering And Technology (Ijciety), Volume 8, Issue 5, Pp. 1220–1227, May 2017.

[19] R. Kalaiprasath, R. Elankavi and Dr. R. Udayakumar. Cloud. Security and Compliance - A Semantic Approach in End to End Security, International Journal Of Mechanical Engineering And Technology (Ijmet), Volume 8, Issue 5, pp-987-994, May 2017.

[20] Thooyamani K.P., Khanaa V., Udayakumar R., Virtual instrumentation based process of agriculture by automation, Middle - East Journal of Scientific Research, v-20, i-12, pp-2604-2612, 2014.

[21] Udayakumar R., Thooyamani K.P., Khanaa, Random projection based data perturbation using geometric transformation, World Applied Sciences Journal, v-29, i-14, pp-19-24, 2014.

[22] Udayakumar R., Thooyamani K.P., Khanaa, Deploying site-to-site VPN connectivity: MPLS Vs IPSec, World Applied Sciences Journal, v-29, i-14, pp-6-10, 2014.

[23] A. Selvakumar., Dr. G. Gunasekaran, "A novel approach in load balancing for dynamic cloud environment using ACO", International Innovative Research Journal of Engineering and Technology, vol. 2, March 2017.

