



Secure distributed deduplication systems with improved reliability

N.Zahira Jahan M.C.A., M.Phil., Associate Professor.,
R.Saraswathi M.C.A., Nandha Engineering College, Erode.

Abstract—Secure distributed Deduplication is a technique for removing duplicate copies of data with more reliability, extensively used in cloud storage to maintain the storage space as free and upload bandwidth. Here is only one copy for each file stored in cloud even if such a file is owned by more number of users. Progress storage utilization is takes place in deduplication system while reducing reliability. Privacy for sensitive data also takes place when they are outsourced by users to cloud. Planning to address the above security test, this paper constructs the first effort to celebrate the idea of scattered reliable deduplication system. This paper recommends a new distributed deduplication systems with upper dependability in which the data chunks are distributed from corner to cornering multiple cloud servers. Safety of the data is done by a secret sharing.

Index Terms—Deduplication, secret sharing, distributed storage system, reliability.

1 INTRODUCTION

Deduplication techniques are broadly engaged to backup data and decrease network and storage transparency by notice and eradicate redundancy among data. It is very difficult to maintain all the data. Multiple copies of the same data has been stored in the cloud and the same content can be accessed by more people at the same time. Data deduplication is used to recover all the problems while using the same data. Storage utilization can also be done by using this technique. Based on various deduplication scheme such as client-side or server-side deduplication, file-level or block-level deduplications, many deduplication systems have been projected. While using the cloud storage it is very special and gorgeous and essential for manage the ever increasing data. Some examples are,

[i] Google Drive, Drop box have been pertaining deduplication to save the network bandwidth and the storage cost with client-side deduplication.

There are two types of deduplication based on the size:

(a) Block-level deduplication- it finds out and eliminate redundancies among data blocks.

(b)File-level deduplication- it determines redundancies between different files and eradicate these redundancies to decrease ability demands. The file can be separated into lesser fixed-size.

[ii] Deduplication method can accumulate the storage space for the cloud storage service providers. Consistency of the system has been decreased by using this. Here only one copy for each file is stored so data consistency is more important.. If such a file was lost, an excessively large

amount of data becomes unreachable because of the unavailability of all the files that share this file. If the value of a file were calculated in terms of the amount of file data that would be lost in case of behind a single chunk, then the quantity of user data lost when a file in the storage system is spoiled grows with the number of the unity of the chunk. Thus, how to assurance of high data consistency in deduplication system is a vital problem. Most of the preceding deduplication scheme has only been measured in a single-server location. on the other hand, as lots of deduplication systems and cloud storage systems are planned by users and function for higher dependability, particularly in archival storage systems where data are vital and should be potted over long time point. This involve that the deduplication storage systems provide reliability comparable to other high-available systems.

Furthermore, the problem for data privacy also arises as more and more sensitive data are being outsourced by users to cloud. Encryption mechanisms have usually been utilized to protect the confidentiality before outsourcing data into cloud. Most commercial storage service provider is reluctant to apply encryption over the data because it makes deduplication impossible. The reason is that the traditional encryption mechanisms, including public key encryption and symmetric key encryption, require different users to encrypt their data with their own keys. As a result, identical data copies of different users will lead to different ciphertext. To overcome all the problems of confidentiality and deduplication, the notion of convergent encryption has been pro-posed and widely adopted to enforce data confidentiality while realizing deduplication. However,

these systems achieved confidentiality of outsourced data at the cost of decreased error resilience. Therefore, how to protect both confidentiality and reliability while achieving deduplication in a cloud storage system is still a challenge.

2. AN OVERVIEW:

In the proposed system, designing methods for a secure deduplication system was specified with high reliability. This introduces the distributed cloud storage servers into deduplication systems to provide better fault tolerance. The secret sharing scheme is used for protect the data confidentiality, which is also compatible with the distributed storage systems. A file is first split and encoded into fragments by using the technique of secret sharing, instead of encryption mechanisms. Here the shares will be distributed across multiple independent storage servers. To support deduplication, a short cryptographic hash value of the content will also be computed and sent to each storage server as the fingerprint of the fragment stored at each server. Only the person who first uploads the data is required to compute and distribute such secret shares, while all following users who own the same data copy do not need to compute and store these shares any more. To recover data copies, users must access a minimum number of storage servers through authentication and obtain the secret shares to reconstruct the data. That is, the secret shares of data will only be accessible by the authorized users who own the corresponding data copy.

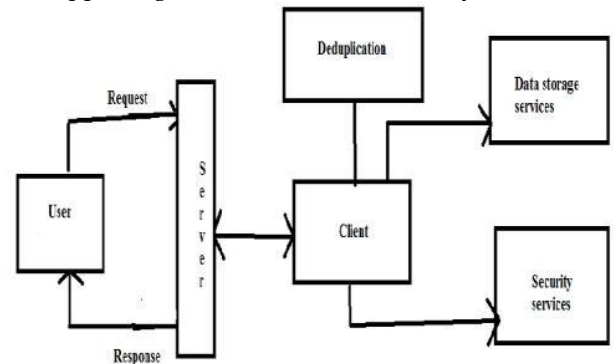
Another feature of proposal is that data integrity, including tag consistency. In the earlier days deduplication methods cannot be directly extended and applied in distributed and multi-server systems. If the same short value is stored at a different cloud storage server to support a duplicate check by using a traditional deduplication method, it cannot resist the collusion attack launched by multiple servers. Any of the servers can obtain shares of the data stored at the other servers with the same short value as proof of ownership. Tag consistency, which was first formalized to prevent the duplicate/ciphertext replacement attack, is considered in the protocol. It prevents a user from uploading a maliciously-generated ciphertext such that its tag is the same with another honestly-generated ciphertext. A deterministic secret sharing method has been formalized and utilized to achieve this. There is no existing work on secure deduplication can properly address the reliability and tag consistency problem in distributed storage systems.

Following are the contributions done by this paper.

To provide efficient deduplication with high reliability, four new secure deduplication systems are proposed for file-level and block-level deduplication respectively. The secret splitting technique, instead of traditional encryption methods, is utilized to protect data confidentiality. Data are

split into fragments by using secure secret sharing schemes and stored at different servers. This proposed constructions support both file-level and block-level deduplication. Security analysis demonstrates that the proposed deduplication systems are secure in terms of the definitions specified in the proposed security model. On the other words confidentiality, reliability and integrity can be achieved in the proposed system. Two kinds of problem attacks are considered in solutions. These are the collusion attack on the data and the collusion attack against servers. In particular, the data remains secure even if the adversary controls a limited number of storage servers.

Ramp secret sharing scheme is implemented in the deduplication system that enables high reliability and confidentiality levels. The result demonstrates that the new proposed constructions are efficient and the redundancies are optimized and comparable with the other storage system supporting the same level of reliability.



System architecture

3. SECURE DISTRIBUTED DEDUPLICATION SYSTEMS:

The future aim of secure distributed deduplication systems is to reliably store data in the cloud while achieving privacy and consistency. The main objective is to allow deduplication and distributed storage of the data diagonally multiple storage servers. As an alternative encrypting the data to keep the privacy of the data, new structures put on the top-secret intense technique to split data into shards.

3.1 The File-level Distributed Deduplication System

To keep efficient duplicate check, tags for each file will be calculated and are directed to S-CSPs. To avoid a conspiracy attack hurled by the S-CSPs, the tags deposited at different storage servers are computationally autonomous and different. The details of the structure are given as follows.

System setup:

The number of Storage servers S-CSPs is expected to be i with identities denoted by id_1, id_2, \dots, id_n , correspondingly in this structure. Describe the security parameter as 1 and set a secret sharing scheme $SS = (\text{Share}, \text{Recover})$, and a tag generation algorithm TagGen . The file storage system for the storage server is set to be $\#.$ File

Upload. To upload a file F , the user relates with S-CSPs to achieve the deduplication. More exactly, the user firstly calculates and sends the file tag $\varphi F = \text{TagGen}(F)$ to S-CSPs for the file duplicate check. When a duplicate is found, the user calculates and sends $\varphi F;idj = \text{TagGen}'(F, idj)$ to the j -th server with identity idj via the secure channel for $1 \leq j \leq n$. The motive for presenting an index j is to avoid the server from receiving the shares of other S-CSPs for the same file or block, which will be described in detail in the security analysis. If $X F;idj$ equals the metadata stored with $X F$, the user will be provided a pointer for the shard stored at server idj .

Else, if no duplicate is found, the user will continue as follows. He runs the secret sharing algorithm SS over F to get $\{c_j\} = \text{Share}(F)$, where c_j is the j -th shard of F . He also calculates $X F;idj = \text{TagGen}'(F, idj)$, which helps as the tag for the j -th S-CSP. As a final point, the user uploads the set of values $\{\varphi F, c_j, X F;idj\}$ to the S-CSP with identity idj via a secure channel. The S-CSP stores these values and returns a pointer back to the user for local storage.

File Download. To download a file F , the user first downloads the secret shares $\{c_j\}$ of the file from k out of n storage servers. Exactly, the user sends the pointer of F to k out of n S-CSPs. After meeting enough shares, the user reconstructs file F by using the algorithm of $\text{Recover}(\{c_j\})$. This method provides fault tolerance and lets the user to remain available even if any limited subsets of storage servers fail.

3.2. The Block-level Distributed Deduplication System

We demonstrate how to attain the fine-grained block-level distributed deduplication. In a block-level deduplication system, the user also needs to firstly achieve the file-level deduplication before uploading his file. If no duplicate is found, the user splits this file into blocks and does block-level deduplication. The system arrangement is the same as the file-level deduplication system; excluding the block size parameter will be defined in addition. Following, the details of the algorithms of File Upload and File Download are mentioned.

File Upload. To upload a file F , the user first achieves the file-level deduplication by sending φF to the storage servers. If a duplicate is found, the user will achieve the file-level deduplication, else, if no duplicate is found, the user achieves the block-level deduplication as follows. Initially divides F into a set of fragments $\{A_i\}$ (where $i = 1, 2, \dots$). For each fragment A_i , the user will achieve a block-level duplicate check by computing $X B_i = \text{TagGen}(A_i)$, where the data handling and duplicate check of block-level deduplication is the same as that of file-level deduplication if the file F is substituted with block B_i . Upon getting block tags $\{X B_i\}$, the server with identity idj computes a block signal vector $R B_i$ for each i . i) If $R B_i = 1$, the user additionally computes and sends $X B_i;idj = \text{TagGen}'(B_i, idj)$ to the S-CSP with

identity idj . If it also equals the matching tag stored, S-CSP sends a block pointer of B_i to the user. At that time, the user keeps the block pointer of B_i and does not need to upload B_i .

ii) If $R B_i = 0$, the user runs the secret sharing algorithm SS over B_i and gets $\{c_{ij}\} = \text{Share}(B_i)$, where c_{ij} is the j -th secret share of B_i . The user also computes $X B_i;idj$ for $1 \leq j \leq n$ and uploads the set of values $\{X F, X F;idj, c_{ij}, X B_i;idj\}$ to the server idj through a secure channel. The S-CSP returns the consistent pointers back to the user. File Download. To download a file $F = \{A_i\}$, the user first downloads the secret shares $\{c_{ij}\}$ of all the blocks A_i in F from k out of n S-CSPs. Exactly, the user sends all the pointers for A_i to k out of n servers. Subsequently gathering all the shares, the user recreates all the fragments A_i using the algorithm of $\text{Recover}(\{c_{ij}\})$ and gets the file $F = \{A_i\}$.

4. SECRET SHARING SCHEME AND BUILDING BLOCKS:

Secret Sharing Scheme has been discussed and a look on two algorithms in a secret sharing scheme, which are Share and Recover. The secret is separated and shared by using Share. With enough shares, the secret can be pulling out and improved with the algorithm of Recover. Here, the Ramp secret sharing scheme (RSSS) is assumed to secretly split a secret into shards. Definitely, the (i, j, p) -RSSS (where $n > j > p \geq 0$) produces n shares from a secret so that,

- (i) The secret can be improved from any j or more shares.
- (ii) No evidence about the secret can be assumed from any p or less shares. Two algorithms, Share and Recover, are defined in the (L, j, p) -RSSS.

Share splits a secret S into $(j - p)$ pieces of equal size, generates p random pieces of the same size, and translates the j pieces using a non-systematic j of- i removal code into i shares of the same size;

Improve takes any j out of i shares as inputs and then outputs the original secret S .

When $p = 0$, the $(i, j, 0)$ -RSSS turn into the (i, j) Rabin's Information Dispersal Algorithm (IDA) [9]. When $p = j - 1$, the $(L, j, j - 1)$ -RSSS becomes the (i, j) Shamir's Secret Sharing Scheme (SSSS) [10].

Tag Generation Algorithm:

There are two kinds of tag generation algorithms are defined. 1. TagGen and 2. TagGen'.

TagGen is the tag generation algorithm that records the original data copy C and outputs a tag $T(C)$. This tag will be produced by the user and practical to achieve the duplicate check with the server. An alternative tag generation algorithm TagGen' precedes as input a file C and an index j and outputs a tag. This tag, generated by users, is used for the proof of ownership for C .

Message authentication code:

A message authentication code (MAC) is a small piece of data used to authenticate a message and to make available integrity and validity assurances on the message. The

message verification code is applied to attain the reliability of the contract out stored files. It can be simply made with a keyed cryptographic hash function. It takes input as a secret key and an arbitrary-length file that supplies to be authenticated, and outputs a MAC. Individual users with the same key making the MAC can confirm the exactness of the MAC value and notice whether the file has been changed or not.

4.1. Advantages of Proposed system:

- i) Distinguishing feature of the proposal is that data integrity, as well as tag consistency, can be achieved.
- ii) There is no current work on safe deduplication can appropriately address the reliability and tag consistency problem in distributed storage systems.
- iii) The both file-level and block-level deduplication are done here.

The proposed deduplication systems are safe in terms of the definitions stated in the proposed security model. If we want to elaborate we can also say that confidentiality, reliability and integrity can be achieved in the proposed system. Two kinds of problem attacks are measured in our solutions. These are the collusion attack on the data and the collusion attack against servers.

The data remains secure even if the opponent controls a limited number of storage servers. The implementation of deduplication systems using the Ramp secret sharing scheme allows high reliability and confidentiality levels. The evaluation results prove that the proposed constructions are efficient and the redundancies are optimized and similar with the other storage system supporting the same level of dependability.

In this system, there are two types of attackers for the data.

- i) Inside attackers
- ii) Outside attackers

All these problems are discovered and managed by constructing the Dekey and secret sharing scheme. Nowadays in the cloud storage this technique is used widely with reliability and consistency.

5. CONCLUSION:

Distributed deduplication systems in high level are to increase the consistency of data however attaining the privacy of the user's outsourced data without an encryption appliance. Improvement on security using tag consistency and integrity were attained. The implementation of deduplication systems using the Ramp secret sharing scheme here gives the demonstration that it acquires small encoding/decoding overhead compared to the network transmission overhead in regular download /upload operations.

REFERENCES

- [1] J. Gantz and D. Reinsel, "The digital universe in 2020: Bigdigital shadows and biggest growth in the forecast," <http://www.emc.com/collateral/analyst-reports/idcthe-digital-universe-in-2020.pdf>, Dec 2012.
- [2] M. O. Rabin, "Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech.Rep. Tech. Report TR-CSE-03-01, 1981.
- [3] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in ICDCS, 2002, pp. 617–624.
- [4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in USENIX Security Symposium, 2013.
- [5] "Message-locked encryption and secure deduplication," in EUROCRYPT, 2013, pp. 296–312.
- [6] 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.
- [7] A. D. Santis and B. Masucci, "Multiple ramp schemes," IEEE Transactions on Information Theory, vol. 45, no. 5, pp. 1720–1728, Jul. 1999.
- [8] 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.
- [9] A. Shamir, "How to share a secret," Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.
- [10] J. S. Plank and L. Xu, "Optimizing Cauchy Reed-solomon Codes for fault-tolerant network storage applications," in NCA-06: 5th IEEE International Symposium on Network Computing Applications, Cambridge, MA, July 2006.
- [11] C. Liu, Y. Gu, L. Sun, B. Yan, and D. Wang, "R-admad: High reliability provision for large-scale de-duplication archival storage systems," in Proceedings of the 23rd international conference on Supercomputing, pp. 370–379.