Secure Distributed and Auditing Reliability for Deduplication

Soumya S. M.[1], **Sarvamangala D. R.**[2]

[1]School of Computing and Information Technology, M. Tech, Reva University, Bangalore, India

[2]School of Computing and Information Technology, Asst. Professor, Reva University, Bangalore, India

Abstract: ***Data deduplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. However, there is only one copy for each file stored in cloud even if such a file is owned by a huge number of users. As a result, deduplication system improves storage utilization while reducing reliability. Furthermore, the challenge of privacy for sensitive data also arises when they are outsourced by users to cloud. Aiming to address the above security challenges, this paper makes the first attempt to formalize the notion of distributed reliable deduplication system. We propose new distributed deduplication systems with higher reliability in which the data chunks are distributed across multiple cloud servers. The security requirements of data confidentiality and tag consistency are also achieved by introducing a deterministic secret sharing scheme in distributed storage systems, instead of using convergent encryption as in previous deduplication systems. Security analysis demonstrates that our deduplication systems are secure in terms of the definitions specified in the proposed security model. As a proof of concept, we implement the proposed systems and demonstrate that the incurred overhead is very limited in realistic environments.***

Keywords: Data deduplication, reliability, distributed deduplication, cloud servers, security requirements, data confidentiality, and tag.

1. Introduction

With the explosive growth of digital data, deduplication techniques are widely employed to backup data and minimize network and storage overhead by detecting and eliminating redundancy among data. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy.

Deduplication has received much attention from both academia and industry because it can greatly improves storage utilization and save storage space, especially for the applications with high deduplication ratio such as archival storage systems. A number of deduplication systems have been proposed based on various deduplication strategies such as client-side or server-side deduplications, file-level or block-level deduplications.

A brief review is given in Section 6. Especially, with the advent of cloud storage, data deduplication techniques become more attractive and critical for the management of ever-increasing volumes of data in cloud storage services which motivates enterprises and organizations to outsource data storage to third-party cloud providers, as evidenced by many real-life case studies. According to the analysis report of IDC, the volume of data in the world is expected to reach 40 trillion gigabytes in 2020. Today's commercial cloud storage services, such as Dropbox, Google Drive and Mozy, have been applying deduplication to save the network bandwidth and the storage cost with client-side deduplication. There are two types of deduplication in terms of the size: (i) *file-level deduplication*,

Which discovers redundancies between different files and removes these redundancies to reduce capacity demands, and (ii) *blocklevel deduplication*, which discovers and removes redundancies between data blocks? The file can be divided into smaller fixed-size or variable-size blocks. Using fixed size blocks simplifies the computations of block boundaries, while using variable-size blocks (e.g., based on Rabin fingerprinting) provides better deduplication efficiency. Though deduplication technique can save the storage space for the cloud storage service providers, it reduces the reliability of the system.

Data reliability is actually a very critical issue in a deduplication storage system because there is only one copy for each file stored in the server shared by all the owners. If such a shared file/chunk was lost, a disproportionately large amount of data becomes inaccessible because of the unavailability of all the files that share this file/chunk. If the value of a chunk were measured in terms of the amount of file data that would be lost in case of losing a single chunk, then the amount of user data lost when a chunk in the storage system is corrupted grows with the number of the commonality of the chunk. Thus, how to guarantee high data reliability in deduplication system is a critical problem.

Most of the previous deduplication systems have only been considered in a single-server setting. However, as lots of deduplication systems and cloud storage systems are intended by users and applications for higher reliability, especially in archival storage systems where data are critical and should be preserved over long time periods. This requires that the deduplication storage systems provide reliability comparable to other high-available systems. Furthermore, the challenge 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 cipher texts. To solve the problems of confidentiality and deduplication, the notion of convergent encryption has been proposed 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. Related Work

Side channels in cloud services: Deduplication in cloud storage [1] Cloud storage services commonly use deduplication, which eliminates redundant data by storing only a single copy of each file or block. Deduplication reduces the space and bandwidth requirements of data storage services, and is most effective when applied across multiple users, a common practice by cloud storage offerings. We study the privacy implications of cross-user deduplication. We demonstrate how deduplication can be used as a side channel which reveals information about the contents of files of other users. In a different scenario, deduplication can be used as a covert channel by which malicious software can communicate with its control center, regardless of any firewall settings at the attacked machine. Due to the high savings offered by cross-user deduplication, cloud storage providers are unlikely to stop using this technology. We therefore propose simple mechanisms that enable cross-user deduplication while greatly reducing the risk of data leakage.

*I*OPS [2] Offline Patching Scheme for the Images Management in a Secure Cloud Environment Recent years have witnessed the development of Cloud Computing. The management of images is a big problem in virtualized environment because there are quantities of Virtual Machine images being stored in a Cloud and most of them are outdated. How to detect the outdated images and patch them efficiently? In this paper, we present a prototype called OPS-Offline Patching Scheme for the Images Management in a Secure Cloud Environment. In OPS, we can detect out the outdated image quickly by a module called Collector. Then a module called Patcher will patch the outdated images. In order to patch an image efficiently, offline patching technology is considered. For the large number of images in the Cloud, parallel scheme is also used. Our experiment results show that OPS can update numerous images efficiently.

3. Contributions

We show how to design secure deduplication systems with higher reliability in cloud computing. We introduce the distributed cloud storage servers into deduplication systems to provide better fault tolerance. To further protect data confidentiality, the secret sharing technique is utilized, which is also compatible with the distributed storage systems. In more details, a file is first split and encoded into fragments by using the technique of secret sharing, instead of encryption mechanisms. These shares will be distributed across multiple independent storage servers. Furthermore, 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 data owner 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. In other words, the secret shares of data will only be accessible by the authorized users who own the corresponding data copy.

To achieve this, a deterministic secret sharing method has been formalized and utilized. To our knowledge, no existing work on secure deduplication can properly address the reliability and tag consistency problem in distributed storage systems. This paper makes the following contributions.

- Four new secure deduplication systems are proposed to provide efficient deduplication with high reliability for file-level and block-level deduplication, respectively. The secret splitting technique, instead of traditional encryption methods, is utilized to protect data confidentiality. Specifically, data are split into fragments by using secure secret sharing schemes and stored at different servers. Our proposed constructions support both file-level and block-level deduplications.
- Security analysis demonstrates that the proposed deduplication systems are secure in terms of the definitions specified in the proposed security model. In more details, confidentiality, reliability and integrity can be achieved in our proposed system. Two kinds of collusion attacks are considered in our 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.
- We implement our deduplication systems using the Ramp secret sharing scheme that enables high reliability and confidentiality levels. Our evaluation results demonstrate 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.

4. Problem Formulation

4.1 System Model

This section is devoted to the definitions of the system model and security threats. Two kinds entities will be involved in this deduplication system, including the user and the storage cloud service provider (S-CSP). Both client-side deduplication and server-side deduplication are supported in our system to save the bandwidth for data uploading and storage space for data storing.

- *User*. The user is an entity that wants to outsource data storage to the S-CSP and access the data later. In a storage system supporting deduplication, the user only uploads unique data but does not upload any duplicate data to save the upload bandwidth. Furthermore, the fault tolerance is required by users in the system to provide higher reliability.

- *S-CSP.* The S-CSP is an entity that provides the outsourcing data storage service for the users. In the deduplication system, when users own and store the same content, the S-CSP will only store a single copy of these files and retain only unique data. A deduplication technique, on the other hand, can, reduce the storage cost at the server side and save the upload bandwidth at the user side. For fault tolerance and confidentiality of data storage, we consider a quorum of S-CSPs, each being an independent entity. The user data is distributed across multiple S-CSPs.

We deploy our deduplication mechanism in both file and block levels. Specifically, to upload a file, a user first performs the file-level duplicate check. If the file is a duplicate, then all its blocks must be duplicates as well, otherwise, the user further performs the block level duplicate check and identifies the unique blocks to be uploaded. Each data copy (i.e., a file or a block) is associated with a *tag* for the duplicate check. All data copies and tags will be stored in the S-CSP.

4.2 Threat Model and Security Goals

Two types of attackers are considered in our threat model: (i) An outside attacker, who may obtain some knowledge of the data copy of interest via public channels. An outside attacker plays the role of a user that interacts with the S-CSP; (ii) An inside attacker, who may have some knowledge of partial data information such as the cipher text. An insider attacker is assumed to be honest-but-curious and will follow our protocol, which could refer to the S-CSPs in our system. Their goal is to extract useful information from user data. The following security requirements, including confidentiality, integrity, and reliability are considered in our security model.

Confidentiality: Here, we allow collusion among the SCSPs. However, we require that the number of colluded S-CSPs is not more than a predefined threshold. To this end, we aim to achieve data confidentiality against collusion attacks.

Integrity: Two kinds of integrity, including tag consistency and message authentication, are involved in the security model. Tag consistency check is run by the cloud storage server during the file uploading phase, which is used to prevent the duplicate/cipher text replacement attack.

Reliability: The security requirement of reliability in deduplication means that the storage system can provide fault tolerance by using the means of redundancy. In more details, in our system, it can be tolerated even if a certain number of nodes fail. The system is required to detect and repair corrupted data and provide correct output for the users.

5. The Distributed Deduplication Systems

The distributed deduplication systems' proposed aim is to reliably store data in the cloud while achieving confidentiality and integrity. Its main goal is to enable deduplication and distributed storage of the data across multiple storage servers. Instead of encrypting the data to keep the confidentiality of the data, our new constructions utilize the secret splitting technique to split data into shards. These shards will then be distributed across multiple storage servers.

5.1 Building Blocks

5.1.1 Secret Sharing Scheme: There are two algorithms in a secret sharing scheme, which are Share and Recover. The secret is divided and shared by using Share. With enough shares, the secret can be extracted and recovered with the algorithm of Recover. In our implementation, we will use the Ramp secret sharing scheme (RSSS) , to secretly split a secret into shards. Specifically, the m(n, k, r)-RSSS (where $n > k > r \geq 0$) generates n shares from a secret so that (i) the secret can be recovered from any k or more shares, and (ii) no information about the secret can be deduced from any r or less shares. Two algorithms, Share and Recover, are defined in the (n, k, r)-RSSS.

- Share divides a secret S into ($k - r$) pieces of equal size, generates r random pieces of the same size, and encodes the k pieces using a non-systematic k-of-n erasure code into n shares of the same size;
- Recover takes any k out of n shares as inputs and then outputs the original secret S. It is known that when $r = 0$, the (n, k, 0)-RSSS becomes the (n, k) Rabin's Information Dispersal Algorithm
- (IDA) . When $r = k-1$, the ($n, k, k-1$)-RSSS becomes the (n,k) Shamir's Secret Sharing Scheme (SSSS)

Tag Generation Algorithm. In our constructions below, two kinds of tag generation algorithms are defined, that is, TagGen and TagGen'. TagGen is the tag generation algorithm that maps the original data copy F and outputs a tag $T(F)$. This tag will be generated by the user and applied to perform the duplicate check with the server. Another tag generation algorithm TagGen' takes as input a file F and an index j and outputs a tag. This tag, generated by users, is used for the proof of ownership for F.

Message authentication code. A message authentication code (MAC) is a short piece of information used to authenticate a message and to provide integrity and authenticity assurances on the message. In our construction, the message authentication code is applied to achieve the integrity of the outsourced stored files. It can be easily constructed with a keyed (cryptographic) hash function, which takes input as a secret key and an arbitrary-length file that needs to be authenticated, and outputs a MAC. Only users with the same key generating the MAC can verify the correctness of the MAC value and detect whether the file has been changed or not.

5.2 The File-level Distributed Deduplication System

To support efficient duplicate check, tags for each file will be computed and are sent to S-CSPs. To prevent a collusion attack launched by the S-CSPs, the tags stored at different storage servers are computationally independent and different. We now elaborate on the details of the construction as follows.

System setup: In our construction, the number of storage servers S-CSPs is assumed to be *n* with Identities denoted by *id*1*, id*2*, · · · , idn*, respectively. Define the security parameter as 1_ and initialize a secret sharing scheme SS = (Share*,* 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 interacts with S-CSPs to perform the deduplication. More precisely, the user firstly computes and sends the file tag *ϕF* = TagGen (*F*) to S-CSPs for the file duplicate check.

File Download: To download a file *F*, the user first downloads the secret shares *{cj}* of the file from *k* out of *n* storage servers. Specifically, the user sends the pointer of *F* to *k* out of *n* S-CSPs. After gathering enough shares, the user reconstructs file *F* by using the algorithm of Recover (*{cj}*). This approach provides fault tolerance and allows the user to remain accessible even if *any* limited subsets of storage servers fail.

6. The Block-level Distributed Deduplication System

In this section, we show how to achieve the fine-grained block-level distributed deduplication. In a block-level deduplication system, the user also needs to firstly perform the file-level deduplication before uploading his file. If no duplicate is found, the user divides this file into blocks and performs block-level deduplication. The system setup is the same as the file-level deduplication system, except the block size parameter will be defined additionally. Next, we give the details of the algorithms of File Upload and File Download.

File Upload: To upload a file *F*, the user first performs the file-level deduplication by sending *ϕF* to the storage servers. If a duplicate is found, the user will perform the file-level deduplication, such as that in Otherwise, if no duplicate is found, the user performs the block-level deduplication as follows. He firstly divides *F* into a set of fragments *{Bi}* (where *i* = 1*,* 2*, · · ·*). For each fragment *Bi*, the user will perform a block-level duplicate check by computing *ϕBi* = TagGen(*Bi*), where the data processing and duplicate check of block-level deduplication is the same as that of file-level deduplication if the file *F* is replaced with block *Bi*.

File Download: To download a file *F* = *{Bi}*, the user first downloads the secret shares *{cij}* of all the blocks *Bi* in *F* from *k* out of *n* S-CSPs. Specifically, the user sends all the pointers for *Bi* to *k* out of *n* servers. After gathering all the shares, the user reconstructs all the fragments *Bi* using the algorithm of Recover (*{·}*) and gets the file *F* = *{Bi}*.

7. Further Enhancement

7.1 Distributed Deduplication System with Tag Consistency

In this section, we consider how to prevent a duplicate faking or maliciously-generated cipher text replacement attack. A security notion of tag consistency has been formalized for this kind of attack. In a deduplication storage system with tag consistency, it requires that no adversary is able to obtain the same tag from a pair of different messages with a non-negligible probability. This provides security guarantees against the duplicate faking attacks in which a message can be undetectably replaced by a fake one. In the previous related work on reliable deduplication over encrypted data, the tag consistency cannot be achieved as the tag is computed by the data owner from underlying data files, which cannot be verified by the storage server.

7.1.1 Deterministic Secret Sharing Schemes

We formalize and present two new techniques for the construction of the deterministic secret sharing schemes. For simplicity, we present an example based on traditional Shamir's Secret Sharing scheme. The description of (*k, n*)-threshold in Shamir's secret sharing scheme is as follows. In the algorithm of Share, given a secret *α ∈ Zp* to be shared among *n* users for a prime *p*, choose at random a (*k* − 1)-degree polynomial function *f*(*x*) = *a*0 + *a*1*x* + *a*2*x*2 + · · · + *ak*−1*xk*−1 *∈ Zp*[*X*] such that *α* = *f*(0). The value of *f*(*i*) *mod p* for 1 ≤ *i* ≤ *n* is computed as the *i*-th share. In the algorithm of Recover, Lagrange interpolation is used to compute *α* from any valid *k* shares.

7.1.2 The First Method

Share. To share a secret *α ∈ Zp*, it chooses at random a (*k* − 1)-degree polynomial function *f*(*x*) = *a*0 + *a*1*x* + *a*2*x*2 + · · · + *ak*−1*xk*−1 *∈ Zp*[*X*] such that *α* = *f*(0), *ai* = *H*(*α*∥ *i*) and *p* is a prime, where *H*(·) is a hash function. The value of *f*(*i*) *mod p* for 1 ≤ *i* ≤ *n* is computed as the *i*-th share and distributed to the corresponding owner. Recover. The description of algorithm Recover is the same with the traditional Shamir's secret sharing scheme by using Lagrange interpolation. The secret *α* can be recovered from any valid *k* shares.

7.1.3 The Second Method

Obviously, the first method of deterministic secret sharing cannot prevent brute-force attack if the file is predictable. Thus, we show how to construct another deterministic secret sharing construction method to prevent the brute-force attack. Another entity, called key server, is introduced in this method, who is assumed to be honest and will not collude with the cloud storage server and other outside attackers.

8. Experiment

We describe the implementation details of the proposed distributed deduplication systems in this section. The main tool for our new deduplication systems is the Ramp secret sharing scheme (RSSS). The shares of a file are shared across multiple cloud storage servers in a secure way.

The efficiency of the proposed distributed systems are mainly determined by the following three parameters of *n*, *k*, and *r* in RSSS. In this experiment, we choose 4KB as the default data block size, which has been widely adopted for block-level deduplication systems. We choose the hash function SHA-256 with an output size of 32 bytes. We implement the RSSS based on the Jerasure Version 1.2. We choose the erasure code in the (*n, k, r*)-RSSS whose generator matrix is a Cauchy matrix for the data encoding and decoding. The storage blowup is determined by the parameters n, k, r. In more details, this value is $n\ k{-}r$ in theory.

- Case 1: $r = 1$, $k = 2$, and $3 \leq n \leq 8$ (Figure 3(a));
- Case 2: $r = 1$, $k = 3$ and $4 \leq n \leq 8$ (Figure 3(b));
- Case 3: $r = 2$, $k = 3$, and $4 \leq n \leq 8$ (Figure 3(c));
- Case 4: $r = 2$, $k = 4$, and $5 \leq n \leq 8$ (Figure 3(d)).

4KB data block) are always in the order of microseconds, and hence are negligible compared to the data transfer performance in the Internet setting. We can also observe that the encoding time is higher than the decoding time. The reason for this result is that the encoding operation always involves all n shares, while the decoding operation only involves a subset of $k < n$ shares.

9. Related Work

Reliable Deduplication systems Data deduplication techniques are very interesting techniques that are widely employed for data backup in enterprise environments to minimize network and storage overhead by detecting and eliminating redundancy among data blocks. There are many deduplication schemes proposed by the research community.

Convergent encryption: Convergent encryption ensures data privacy in deduplication. Bellare et al. Formalized this primitive as message-locked encryption, and explored its application in space efficient secure outsourced storage. There are also several implementations of convergent implementations of different convergent encryption variants for secure deduplication.

Proof of ownership*:* Harnik et al. Presented a number of attacks that can lead to data leakage in a cloud storage system supporting client-side deduplication. To prevent these attacks, Halevi et al. proposed the notion of ―proofs of ownership" (PoW) for deduplication systems, so that a client can efficiently prove to the cloud storage server that he/she owns a file without uploading the file itself. Several PoW constructions based on the Merkle Hash Tree are proposed [12] to enable client-side deduplication, which includes the bounded leakage setting. Pietro and Sorniotti [23] proposed another efficient PoW scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof. Note that all of the above schemes do not consider data privacy.

PoR/PDP. Ateniese et al. Introduced the concept of proof of data possession (PDP). This notion was introduced to allow a cloud client to verify the integrity of its data outsourced to the cloud in a very efficient way. Juels et al. Proposed the concept of proof of irretrievability (PoR). Compared with PDP, PoR allows the cloud client to recover his outsourced data through the interactive proof with the server. This scheme was later improved by Shacham and Waters [28]. The main difference between the two notions is that PoR uses Error Correction/Erasure Codes to tolerate the damage to portions of the outsourced data.

10. Conclusion

We proposed the distributed deduplication systems to improve the reliability of data while achieving the confidentiality of the users' outsourced data without an encryption mechanism. Four constructions were proposed to support file-level and fine-grained block-level data deduplication. The security of tag consistency and integrity were achieved. We implemented our deduplication systems using the Ramp secret sharing scheme and demonstrated that it incurs small encoding/decoding overhead compared to the network transmission overhead in regular upload/download operations.

References

[1] Amazon, ―Case Studies," https://aws.amazon.com/solutions/casestudies/# backup.
[2] J. Gantz and D. Reinsel, ―The digital universe in 2020: Big data, bigger digi tal shadows, and biggest growth in the far east," http://www.emc.com/collateral/analyst-reports/idcthe- digital-universe-in-2020.pdf, Dec 2012.
[3] M. O. Rabin, ―Fingerprinting by random polynomials," Center for Research in Computing Technology, Harvard University, Tech. Rep. Tech. Report TR-CSE-03-01, 1981.
[4] 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.
[5] M. Bellare, S. Keelveedhi, and T. Ristenpart, ―Dupless: Serveraided encryption for deduplicated storage," in *USENIX Security Symposium*, 2013.
[6] ——, ―Message-locked encryption and secure deduplication," in *EUROCRYPT*, 2013, pp. 296–312.
[7] 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.
[8] A. D. Santis and B. Masucci, ―Multiple ramp schemes," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1720–1728, Jul. 1999.
[9] 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.
[10] A. Shamir, ―How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[11] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, ―Secure deduplication with efficient and reliable convergent key management," in *IEEE Transactions on Parallel and Distributed Systems*, 2014, pp. vol. 25(6), pp. 1615–1625.
[12] 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.
[13] 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.