Scientific Journal of Impact Factor (SJIF): 4.14

e-ISSN : 2348-4470
p-ISSN : 2348-6406

International Journal of Advance Engineering and Research Development

Volume 3,Issue 4,April -2016

Refined distributed replica's with Improved reliability and security

Uttam Pancholi [1], Priyanka Suryavanshi [2], Sneha Phuge [3] and Prof. Mangesh Manake [4]

[1]*Computer Engineering, Savitribai Phule Pune University/D.Y.P.I.E.T., Ambi / Pune, Maharashtra 410506, India*
[2]*Computer Engineering, Savitribai Phule Pune University/D.Y.P.I.E.T., Ambi / Pune, Maharashtra 410506, India*
[3]*Computer Engineering, Savitribai Phule Pune University/D.Y.P.I.E.T., Ambi / Pune, Maharashtra 410506, India*
[4]*Computer Engineering, Savitribai Phule Pune University/D.Y.P.I.E.T., Ambi / Pune, Maharashtra 410506, India*

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

Keywords*- Deduplication, Distributed storage system, Reliability, Secret sharing scheme, File level deduplication, Block level Deduplication.*

I. INTRODUCTION

With explosive growth of digital data, deduplication techniques are mostly employed to backup the data and minimize the network and storage overcome by detecting and eliminating the redundancy among the data. Instead of keeping the multiple data copies with the same content, the deduplication eliminates the redundant data by keeping the only one physical copy and referring other redundant data to that original or physical copy. Deduplication is received much attention from the both of academia and industry because of it greatly improves the storage utilization and save the storage space, especially for applications with the high deduplication ratio such as the archival storage systems. A huge number of deduplication systems have been proposed based on the various deduplication strategies which is file and block-level deduplications, client-side or server-side deduplications. With the advent of cloud storage, data deduplication techniques is become more attractive and critical for management of ever-increasing volumes of the data in the services of cloud storage which is motivates enterprises and organizations to the outsource data storage.

II. LITERATURE SURVEY

A: Reclaiming Space from the Duplicate Files in a Server less Distributed File System

Authors: John R. Douceur

The Far site distributed file system provides the availability by replicating each file onto the multiple desktop computers. Since the this replication consumes significant storage space, it's important to reclaim the used space where is possible. Over measurement of 500 desktop file systems which shows that nearly half of all consumed space is occupied by the duplicate files. We present a mechanism to reclaim the space from this incidental duplication to make it the available for controlled the file replication. In our mechanism includes the

1) Convergent encryption, which is enables duplicate files to coalesced into the space of a one single file, even if the files are encrypted with the different users' keys, and the

2) SALAD, it is Self- Arranging, Lossy, and Associative Database for aggregating file content and the location information in decentralized, scalable, fault-tolerant manner. Large-scale simulation experiments shows that duplicate-file coalescing system is scalable, highly effective, and the fault-tolerant.

B: DupLESS: Server-Aided Encryption for the Deduplicated Storage

Authors: Mihir Bellare

Cloud storage service providers such as a Drop box, Mozy, and many others perform deduplication to save the space by only storing one copy of the each file uploaded. Should they clients conventionally encrypt their files, however, savings lost. Message-locked encryption (the most prominent manifestation of which is the convergent encryption) resolves this type of tension. However it is inherently subject to the brute-force attacks that can recover files falling into the known set. We propose the architecture which provides secure Deduplicated storage resisting in brute-force attacks, and realize it in system which is called DupLESS. In the DupLESS, clients encrypt under message-based keys obtained from key-server

via oblivious PRF protocol. It enables clients to the store encrypted data with the existing service, have service perform deduplication on the their behalf, and yet achieves the strong confidentiality guarantees. We show that the encryption for Deduplicated storage can achieve space savings and performance close to that of using storage service with the plaintext data.

C: CDStore: Toward the Secure, Reliable, and the Cost-Efficient Cloud Storage via Convergent Dispersal

Authors: Mingqiang Li

We present the CDStore, which is disperses users' backup data across a multiple clouds which provides a unified multi cloud storage solution with reliability, security, and also cost- efficiency guarantees. CDStore builds on augmented secret sharing scheme called as a convergent dispersal, which supports the deduplication by using the deterministic content-derived as inputs to the secret sharing. We present the design of CDStore, and in particular, describe the how it combines convergent dispersal with the two-stage deduplication to achieve the both bandwidth and storages a vings and be robust against side-channel attacks. We evaluate performance of our CDStore prototype using the real-world workloads on LAN and the commercial cloud test beds. Our cost analysis is also demonstrates that the CDStore achieves the monetary cost saving of 70 percentage over a baseline cloud storage solution using the state-of-the-art secret sharing.

D: Secure Deduplication and Data Security with the Efficient and Reliable CEKM

Authors: N. O. Agrawal

Secure deduplication is the technique for eliminating the duplicate copies of storage data, and provides the security to them. To reduce the storage space and upload bandwidth in the cloud storage deduplication has been well-known technique. For that purpose the convergent encryption has extensively adopt for the secure deduplication, critical issue of the making convergent encryption practical is that to efficiently and reliably manage a large number of convergent keys. The basic idea in the this paper is that we can eliminate duplicate copies of storage data and limit damage of stolen data if we decrease the value of the that stolen information to the attacker. In this paper, makes first attempt to formally address the problem of achieving the efficient and reliable key management in the secure deduplication. We first introduce a baseline approach in which the each user holds an independent master key for the encrypting the convergent keys and outsourcing them. Whenever, such a baseline key management scheme generates the enormous number of keys with the increasing number of users and requires users to dedicatedly protect that the master keys. To this end, we propose Dekey, User Behavior Profiling and the Decoys technology. Dekey new construction in which the users do not need to manage any kinds keys on their own but instead of the securely distribute the convergent key shares across the multiple servers for insider attacker. As the proof of the concept, we implement the Dekey using the Ramp secret sharing scheme and demonstrate that the Dekey incurs limited overhead in the realistic environments.

III. IMPLEMENTATION

A: Module 1: The Data de-duplication

Data deduplication involves the finding and removing the duplication within the data without compromising its integrity. The goal is to store more data in the less space by segmenting files into the small variable-sized chunks (32–128 KB), identifying the duplicate chunks, and maintaining a single copy of the each chunk. Redundant copies of chunk are replaced by a reference to the one single copy. The chunks are compressed and then organized into the special container files in System Volume Information folder. After the deduplication, files are no longer stored as the independent streams of data, and they are replaced with stubs that point to the data blocks which is stored within a common chunk store. Because of these files share blocks, those blocks are only stored once, which reduces disk space needed to store all files. During the file access, correct blocks are transparently assembled to serve data without calling the application or user having any knowledge of the on-disk transformation to the file. This is enables administrators to apply deduplication to the files without having to worry about any change in the behavior to the applications or impact to users who are accessing the those files.

After a volume is enabled for deduplication and data is optimized, the volume contains the following:

- Unoptimized files: For example, unoptimized files could include the files that do not meet the selected file-age policy setting, system state files, alternate the data streams, encrypted files, files with the extended attributes, files smaller than the 32 KB, and other reparse point files, or files in use by the other applications (the "in use" limit is removed in Windows Server 2012).
- Optimized files: The Files that are stored as the reparse points that contain pointers to map of the respective chunks in chunk store that are needed to restore a file when requested it is.
- Chunk store: Location for optimized file data.
- Additional free space: The optimized files and chunk store occupy the much less space than they did prior to optimization.

B: Module 2: The Distributed de-duplication systems

The distributed systems' proposed aim is to reliably store the data in cloud while achieving the confidentiality and integrity. Its goal is to enable and distributed storage of data across the multiple storage servers. Instead of encrypting data to keep the confidentiality of data, our new constructions utilize the secret splitting technique to split the data into shards. These shards will then be distributed across the multiple storage servers. Building Blocks Secret Sharing Scheme. There are the two algorithms in secret sharing scheme, which is Share and Recover. The secret is shared and divided by using the Share. With enough shares, secret can be recovered and extracted with the help of algorithm of Recover. In the our implementation, we will use Ramp secret sharing scheme (RSSS), to secretly split secret into the shards. Specifically, the (r, k, n)-RSSS (where is $0 \leq r < k < n$) generates the n shares from a secret so that (i) secret can be recovered from the any k or more shares, and (ii) no information about secret can be deduced from the any r or less shares. Two algorithms, Share and Recover, which is defined in the (n, k, r)-RSSS.

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

C: Module 3: The File-level Distributed De-duplication System

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

System setup: In the our construction, the number of storage servers S-CSPs which is assumed to be n with the identities denoted by id1, id2, · · ·, idn, respectively. Define security parameter as 1_ and initialize the secret sharing scheme SS = (Recover, Share), and a tag generation algorithm is TagGen. The file storage system for the storage server is set to be the File Upload. And then to upload the file F, the user interacts with the S-CSPs to perform de-duplication. More precisely, the user firstly computes and then sends the file tag ϕF = TagGen (F) to S-CSPs for file duplicate check.

- If duplicate is found, the user computes and sends the $\phi F; idj$ = TagGen′(F, idj) to the j-th server with identity idj via secure channel for $1 \leq j \leq n$ (which could be implemented by cryptographic hash function $Hj(F)$ related with the index j). The reason for introducing an index j is to prevent server from getting the shares of other S-CSPs for the same file or same block, which will be explained in detail in security analysis. If $\phi F; idj$ matches the metadata stored with the ϕF, the user will be provided a pointer for shard stored at server idj.
- Otherwise, if there is no duplicate is found, the user will proceed as follows. He runs secret sharing scheme algorithm SS over F to get the $\{cj\}$ = Share (F), where cj is the j-th shard of the F. He also computes the $\phi F; idj$ = TagGen′ (idj, F), which is serves as the tag for jth S-CSP. Finally, user uploads the set values of $\{\phi F, cj, \phi F; idj\}$ to the S-CSP with the identity idj via the secure channel. The S-CSP stores these values and returns a pointer back to the user for the local storage.

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

D: Module 4: The Block-level Distributed System

In the this section, we show how to achieve the fine-grained block-level distributed of de-duplication. Then in the block-level de-duplication system, user also needs to first of all perform the file-level de-duplication before uploading the his file. If there is no duplicate is found, the user divides the this file into the blocks or chunks and performs the block-level of de-duplication. The system setup is same as the file-level de-duplication system, except the block size parameter will be defined as a additionally. Then next, we give the details of algorithms of the File Upload and Download. *File Upload*: To upload the file F, user first performs file-level de-duplication by sending ϕF to the storage servers. If the duplicate is found, the user will perform the file-level de-duplication. Otherwise, if there is no duplicate is found, the user performs block-level de-duplication as follows. He firstly divides F into the set of the fragments $\{Bi\}$ (where i *is* = 1, 2, · · ·).

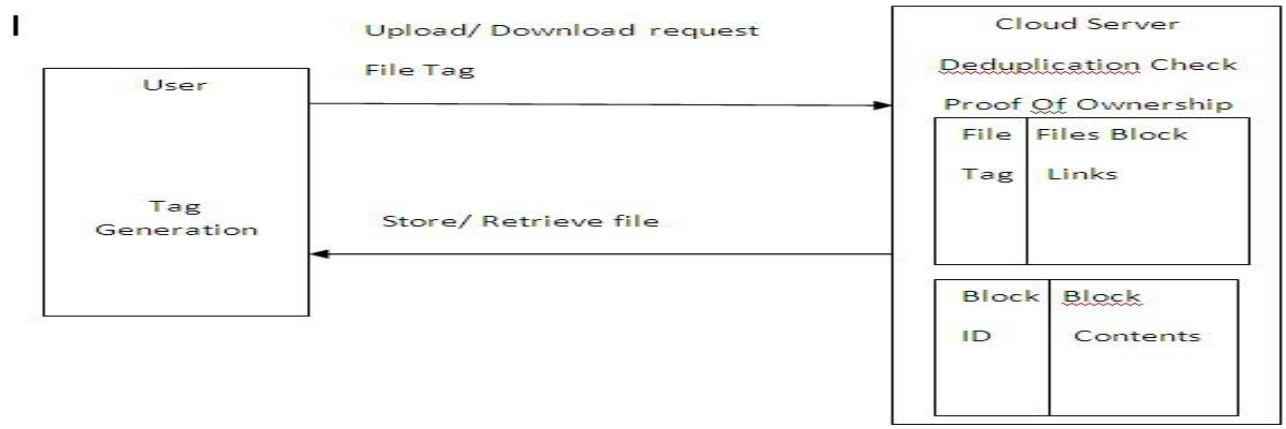
For the each fragment Bi, the user will perform block-level duplicate check by computing the ϕBi = TagGen(Bi), where data processing and duplicate check block-level de-duplication is the same as the that of file-level de-duplication even if the file F is replaced with the block Bi. Upon receiving block tags $\{\phi Bi\}$, the server with the identity of idj computes a block signal vector of σBi for each i.

- i) If σBi=1, the user further computes and sends the $\phi Bi; j$ = TagGen′(Bi, j) to the S-CSP with identity idj . If it also matches the corresponding tag stored, S-CSP returns the block pointer of Bi to user. Then, the user keeps the block pointer of Bi and does not need to upload the Bi.
- ii) If the σBi=0, the user runs secret sharing algorithm over Bi and gets the $\{cij\}$ = Share (Bi), where the cij is the j-th secret share of the Bi. The user also computes the $\phi Bi; j$ for $1 \leq j \leq n$ and uploads set of values $\{\phi F, \phi F; cij, idj, j,$ ϕBi $;\}$ to the server idj via the secure channel. The S-CSP is returns the corresponding pointers back to the user.

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

IV. FIGURES

A: System architecture:



V. EQUATIONS

A: Mathematical Model:

- P- procedure,
- O- Output.
- I-{F,U}
- F-Files set of {F1, F2,…., FN}
- U- No of Users {U1, U2,……,UN}
- Procedure (P):
- P={POW, n ,, , ϕ, i,j ,m , k}
- Where,
- POW - proof of ownership.
- n - No of servers.
- proof of ownership in the blocks.
- proof of ownership in the files
- ϕ - tag.
- i- Fragmentation.
- j- No of server.
- m-message
- k- Key

i) File Upload (FU):

Step 1: File level deduplication

- If there is the file duplicate is found, the user will run PoW protocol POWF with each S-CSP to prove file ownership for the *j*-th server with identity *idj*, the user first computes
- $\phi F;idj = \text{TagGen}'(F, idj)$
- and then runs the PoW proof algorithm with respect to the $\phi F, idj$. If the proof is passed, user will be provided a pointer for piece of file stored at *j*-th S-CSP. Otherwise, if there is no duplicate is found, the user will proceed as follows:
- First divides *F* into the set of fragments *{Bi}* (where $i = 1, 2, \cdots$).
- For each fragment *Bi*, user will perform a block-level duplicate check.

Step 1: Block Level deduplication

- If there is the duplicate in S-CSP, the user runs PoWBon input:
- $\phi Bi;j = \text{TagGen}'(Bi, idj)$
- with server to prove that he owns the block of *Bi*. If it passed, server simply returns the block pointer of *Bi* to user. The user then keeps the block pointer of the *Bi* and does not need to upload the *Bi*.

ii) Proof of ownership (POW):

Step 1: compute and send ϕ' to the verifier.

Step 2: presen which is the proof to the storage server that he owns the *F* in an interactive way with respect to the ϕ' The PoW is successful if proof is correct.

- $\phi' = \phi(F)$

iii) File Download (FD):

To download the file *F*, user first downloads the secret shares *{cij,mfj}* of file from *k* out of *n* storage servers. Specifically, user sends all the pointers for *F* to *k* out of *n* servers. After gathering all the shares, user reconstructs the file *F*, *mac F* by using the algorithm of Recover(*{·}*). Then, he verifies the correctness of these tags to check integrity of the file stored in S-CSPs.

iv) Output (O):

User can upload, download, recover and share files on the cloud server and also provide the data dedupliaction with reliability.

VI. APPLICATIONS

Data deduplication is the technique for eliminating the duplicate copies of data, and has been mostly used in the cloud storage to reduce the storage space and also upload bandwidth.

VII. ADVANTAGES

- By using the block level deduplication reduce storage space
- Reliability issue is overcome
- Distributed Deduplication System is performed.

VIII. DISADVANTAGES

To solve the reliability problem by distributed deduplication system the storage space increase lightly.

IX. POSSIBLE OUTCOMES OR RESULT

The distributed deduplication systems are to improve reliability of data. Four constructions we are proposed to support the file-level and block-level data deduplication. Our deduplication systems using Ramp sharing secret scheme and then demonstrated that it is incurs small encoding and decoding overhead in compared to the network transmission which is overhead in the regular upload and download operations.

Storage administrators are struggling to handle the spiraling volumes of documents just like, audio, images, video and large email attachments. Adding storage is not always best solution, and many companies are turning to the data reduction technologies such as data deduplication. This article explains basic principles of data deduplication, and looks at the implementation issues.

X. CONCLUSIONS

We proposed the distributed de-duplication systems to improve reliability of the data while achieving the confidentiality of the users' out sourced data without the mechanism of the encryption. Four constructions were proposed to support the file-level and also the fine-grained block-level data of the de-duplication. Then the security of the tag consistency and also integrity were achieved. In this we implemented our de-duplication systems using the Ramp secret sharing scheme (RSSS) and demonstrated that it incurs small encoding and decoding overhead in compared to the network transmission overhead in the regular download and upload operations.

REFERENCES

[1] Amazon, "Case Studies," https://aws.amazon.com/solutions/casestudies/# backup.

[2] D. Reinsel and J. Gantz, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East," http://www.emc.com/collateral/analyst-reports/idc the-digital-universe-in-2020.pdf, Dec 2012.

[3] M. O. Rabin, "Fingerprinting by the random polynomials," Center for Research in the Computing Technology, Harvard University, Tech. Rep. Tech. Report TR-CSE-03-01, 1981.

[4] D. Simon, J. R. Douceur, A. Adya, W. J. Bolosky, and M. Theimer, "Reclaiming space from duplicate files in a server less distributed file system." in *ICDCS*, 2002, pp. 617–624.

[5] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Serve raided encryption for Deduplicated storage," in *USENIX Security Symposium*, 2013.

[6] "Message-locked encryption and secure de-duplication," in *EUROCRYPT*, 2013, pp. 296–312.

[7] C. Meadows and G. R. Blakely, "Security of ramp schemes," in *Advances in Cryptology: Proceedings of CRYPTO '84*, ser. Lecture Notes in Computer Science, G. R. Blakely and D. Chaum, Eds. Springer-Verlag Berlin/Heidelberg, 1985, vol. 196, pp. 242–268.

[8] B. Masucci and A. D. Santis, "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 the 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] W. Lou, J. Li, X. Chen, M. Li, J. Li, and P. Lee, "Secure de-duplication with reliable and efficient convergent key management," in *IEEE Transactions on Parallel and Distributed Systems*, 2014, pp. vol. 25(6), pp. 1615–1625.

