IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

Reliable and Secured Distributed Deduplication System in Cloud Computing

Rohit Jagtap[1], Akshay Deore[2], Kiran Mate[3]

Student, Computer Engineering, ARMIET, Asangaon, India [1, 2, 3]

Abstract: Cloud Storage has been commonly used by many users, deduplication is a one of the technique used in such cloud storages. Basic idea of deduplication is that there should be only single instance of a file stored in the servers even if it is used by millions of users. Although deduplication increases storage consumption, the technique itself is not very reliable. Likewise, when the user uploads some file security concern of confidentiality of that file also arises. With the above security concerns keeping in mind we put forward this paper with an idea of creating a reliable distributed deduplication system. With our system we target to improve reliability of files with them scattered across multiple serves. With our new covert sharing mechanism we also meet the security requirements of data confidentiality and consistency, all of this without using any cryptography mechanism of any sort. Our tests show that this system is secured as it fulfills our security threat model. As a proof of concept, we implement the system we put forward and show that the incurred operating cost is very limited in realistic environments.

Keywords: Deduplication, distributed storage system, reliability, covert sharing.

I. INTRODUCTION

With increase in the amount of digital data, deduplication techniques are widely used to reduce network overhead and storage overhead of servers by checking for redundancy and removing them. Deduplication removes redundant files and keeps single instance and refer other redundant copies to that instance of the file, instead of having multiple instance of a single file. Lately deduplication has gathered much attention as it reduces network overhead and reduces storage utilization saving storage spaces.

There are a number of deduplication techniques used such as client-side server-side deduplication, file-level deduplication and block-level deduplication. When cloud storage comes in question data deduplication becomes more important and critical as it manages the ever increasing amount of data in cloud storage services, due this increasing data organizations and enterprises are forced to use third party cloud storages providers. IDC reports shows that by 2020 the amount of data in the world is expected to reach 40 million petabytes that's 40 trillion gigabytes.

There are two types of data duplication based on size:

(i) File-level deduplication: - This focuses on discovering redundancies between different file and eliminate these redundancies to reduce overhead.

(ii) Block-level deduplication: - A file can be decomposed into smaller blocks these blocks can be of variable or fixed sizes. This technique focuses on discovering and eliminating redundancies between these created blocks of files. Using fixed sized blocks makes computations simple and is simple in complexity. Whereas using variable sized blocks is efficient.

While using deduplication techniques can reduce storage overhead it is not very reliable concept. As there is only once copy of file stored on the servers and is used by multiple user the data reliability becomes difficult to achieve. If a file/chunk is lost, a huge amount of data becomes unavailable as the chuck shared by that file/chuck becomes unavailable.

Thus it becomes important to assure a high data reliability in deduplication but how? Previous deduplication systems were mostly focused on single server. However, deduplication system and cloud storages are made for users as it should have higher reliability specifically in archival storage systems as the data is critical and supposed to be stored over a longer period of time. This requires for our system to have more reliability than high-available systems.

Furthermore, the question of data privacy arises as data is being outsourced to cloud. Cryptography is one of the technique used in protecting the confidentiality of the outsourced data. But with encryption mechanisms the deduplication mechanism becomes difficult to implement. Because in traditional encryption mechanisms, it requires different users to convert their files into cipher texts using their own keys. This creates different cipher texts of identical copies of file. To solve this problem of confidentiality with deduplication, an idea of convergent encryption is proposed have deduplication with confidentiality. But this system although make data confidential it also makes the data less resilient to errors. Hence we arise with a question how to achieve deduplication with ensuring confidentiality as well as reliability.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

II. PROBLEM FORMULATION

A. System Model

This section defines our system model and security threat model we want to tackle. There are two entities that will be involved in our system User and Cloud service provider(CSP). To save bandwidth for data uploading and space for storing we support both client-side and server-side deduplication.

User: - A user is an entity who can outsource his data to CSP and can access it later. A user uploads only unique data but does not upload any duplicated data to save upload bandwidth in a deduplication system. Furthermore, to provide high reliability a fault tolerance mechanism is required.

CSP: - The data outsourced by the user is stored in the Cloud Service Provider. Even if the user owns and stores a duplicated content in the storage the CSP will take care of the deduplication and will store a single instance of the file and retain only unique data. A deduplication technique, can save the upload bandwidth as it reduces the storage cost by performing deduplication at server side. The user data is distributed across multiple CSPs.

We perform both file level and block level deduplication on these servers. Specifically, when a file is uploaded first it checks for file level deduplication. If it is a duplicate, then all its blocks must be duplicates as well, otherwise, the file is divided to blocks and check for duplicate blocks and store only the unique blocks only. A tag is associated with each data copy (i.e., a file or a block) for the duplicate check. The CSP store all the tags and data copies.

B. Threat Model and Security Goals

Our threat model covers two types of attackers: (i) An outside attacker, who uses public channels to obtain some knowledge of the data copy. This type of attacker plays the role of user interacting with the CSPs; (ii) An inside attacker, who has some information about the data i.e. partial knowledge of the data such as 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. Our model consists of the following security requirements, Confidentiality, Integrity, Reliability.

Confidentiality: - Here, CSPs allows collusions among them. Although we must look that the number of colluded CPSs should not go beyond the predefined threshold. To this end, we propose to get data confidentiality against collusion attacks. Even if the attacker controls a predefined number of CSPs, the data stored and distributed should remain secure. The attackers goal is to retrieve data which does not belong to him. This also implies that the attacker cannot access the data which he does not own.

Integrity: - Tag consistency and message authentication are two integrity involved in our security model. A duplicate/cipher text replacement is prevented due to the tag consistency check which in run by the cloud storage in file uploading phase. If an attacker tries to upload a maliciously generated cipher text, if the tag mismatch with the honestly generated tag the cloud server can detect such behaviour. Thus making the user carefree that the data is not being replaced or being unable to decrypt. To check whether the downloaded or decrypted data is uncorrupted we perform a message authentication check. This security requirement protects the CSPs from and inside attacker.

Reliability: - The reliability in deduplication is a security measure that can provide fault tolerance by using means of redundancy. In more details, in our system, even a node fail it can be tolerated. The system should provide user with the correct output when the system detects such faults.

III. DISTRIBUTED DEDUPLICATION SYSTEM

We propose a distributed deduplication system, its aim is to achieve data deduplication while achieving data confidentiality, reliability and integrity. The systems main goal is to perform deduplication and store data across distributed cloud servers. We are using secret splitting technique which divides data into shards instead of using conventional encryption technique to achieve confidentiality. These shards will get distributed across multiple distributed servers.

A. Building Blocks

Covert Sharing Scheme: - The covert sharing scheme proposed by us has two algorithms namely Share and Recover. Share algorithm is used to divide the data and share it. When adequate amount of shares is gathered, the Recover algorithm is used to extract and recover the data. With the help of Ramp Secret Sharing Scheme (RSSS) we will be splitting the data into fragments. The data is split into n number of shares and produce (n, k, r) such that $n > k > r \geq 0$ by following the protocols that (i) data can be extracted from any k or more shares, and (ii) anyone cannot deduce any information by having r or less shares.
There are two algorithms in RSSS Share and Recover they are defined as(n, k, r).

Share, let secret be S, the share algorithm splits the data into (k-r) fragments of same sizes. This generates n number of shares in which there are r random pieces and k is generated suing k/n code in shares;
Recover algorithm produces the original data by taking any k piece from the n number of shares.

Tag Generation Algorithm: -This system we have defined two algorithms to generate tags, those are TagGen and TagGen'. Suppose we have F as our unique data instance, TagGen algorithm produces tag T(F) by mapping F. The deduplication check is performed with the help of this user

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

generated tag. With the help of file and an index j TagGen' outputs a tag. This tag gives a proof of ownership of the user to the file F.

Message Authentication Code: - A message authentication code is an information which provides integrity that the message is authentic and unaltered. MAC is used in our system to provide integrity on the outsourced data saved on cloud servers. A keyed cryptographic hash function is used to generate this message authentication code, a secret key and a file that need to be authenticated is given as input to this function, and that function gives MAC as the output. Only user with the same MAC value can authenticate the integrity of the file.

B. File Level Distributed Deduplication System

For the duplication check, for each file to be uploaded tags are generated and sent to CSPs. The tags generated are stored on different servers and are independent, this is done to prevent a collusion attack launched by the CSPs.

System Setup: - In our system, let the number of storage servers in CSPs be n and their identities denoted by s1, s2, · · ·, sn, respectively. Define and initialize the initial security parameter as 1 λ and a secret sharing scheme SS= (Share, Recover) and TagGen as our tag generation algorithm. The file storage system for the storage server is set to be ⊥.

File Upload: - Uploading a file F, the user uploads a file f onto CSPs which performs deduplication. Specifically, the system will apply the TagGen algorithm of the file F i.e. tag φF = TagGen(F) this tag is then send to CSPs for the duplication check

When a duplicate is found, the user calculates φF,sj = TagGen′ (F, sj) and sends it to the j server with sj as its identity via protected network. Index j prevents from other CSPs shares of the same file or block from other CSPs. If φF stored in the jth CSP(sj) matches the metadata of φF,sj, a pointer is provided pointing to the shard stored in the sj.

When a duplicate is not found. A secret sharing algorithm is performed on F to get {cj} = Share(F), where cj is the j-th shard of F. Also computing the TagGen algorithm to calculate φF,sj = TagGen′ (F, sj), to generate a tag for the jth CSPs. Finally, a protected channel is used to upload a the set of values calculated {φF ,cj , φF,sj } with sj as the CPS's identity. A pointer is returned back to the user after the CPS had stored these values for local storage.

File Download: - For downloading a file F, the user will firstly download the shares {cj} of the desired file from k out of n servers. Precisely, the user sends the pointer of F to k out of n CSPs. After enough shares had been gathered the user will reconstruct the file F using the Recover Algorithm Recover({cj}).

Even if the some of the storage servers fails the user can still access his file using above method this provides a fault tolerance mechanism.

C. Block Level Distributed Deduplication System

In a block-level deduplication system, before uploading his file the user needs to perform a file level deduplication. When no duplicates are found the user will further divide the file into chunks and perform block level deduplication on every chunk of that file. Except for the additional block size parameter, the system setup is same as that of file level deduplication.

File Upload: - Uploading a file F, Primarily the user performs a file level deduplication and send the φF to the storage servers. When it is a duplicate, file level deduplication is performed by the user. Otherwise, when there is no duplicate found on the server, the user will perform a block level deduplication.

The user will primarily divide the file into chunks like a set of fragments {Bi} (where i = 1, 2, · · ·). After generating the fragments, the user will perform block level deduplication of on each fragment {Bi} by calculating φBi = TagGen(Bi), by replacing file F with block Bi in the file level deduplication the data processing and the duplication check is same as that of the file level deduplication.

A block signal vector σBi is computed for each i by the server sj when the block tags {φBi} are received.

i) When σBi=1, the user calculates φBi,j = TagGen′ (Bi, j) and sends it to the CSP with sj as its identity. A block pointer of Bi is returned to the user when the tag is matched with a tag stored. If a user receives a block pointer of the block, then he does not need to upload that block Bi.

ii) When σBi=0, a Share is generated such as {cij} = Share(Bi) (where cij is the jth SS of Bi) by performing a secret sharing algorithm on Bi. After calculating φBi,j the user uploads the set of values {φF , φF,idj , cij , φBi,j} to the sj server using a protected channel.

File Download: - Downloading a file F={Bi}, the user must acquire the shares {cij} of all the blocks Bi which are of file F from k of n CSPs. Precisely, k out of n servers receives all the block pointer Bi sent by the user. After all the shares have been gathered Recover {{.}} algorithm is applied on all the shares to reconstruct the Bi and finally reconstruct the file F using the acquired fragments Bi such that F={Bi}.

IV. CONCLUSION

We propose a distributed deduplication system which focuses on improving reliability of data and also achieving confidentiality of users' saved data all the while performing deduplication. We proposed file-level deduplication, block-level deduplication and client-side as well as server-side deduplication. We achieved integrity with the help of tag consistency and Machine Authentication Code (MAC). We used cover secret sharing mechanism and show that our system reduces network overhead and storage overhead while uploading and downloading.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

ACKNOWLEDGMENT

This work was supported by Alamuri Ratnamala Institute of Engineering and Technology. A big thank you to our professor and guide **Prof. Ravi Raju Y** for guiding us throughout this research.

REFERENCES

[1] Walter Santos, Thiago Teixeira, Carla Machado, Wagner Meira Jr., Altigran S. Da Silva, Renato Ferreira, DorgivalGuedes, "A Scalable Parallel Deduplication Algorithm".
[2] Jingwei Li, Jin Li, DongqingXie and Zhang Cai, "Auditing and Deduplicating Data in Cloud".
[3] Ronald L. Krutz, Russell Dean Vines, "Cloud Security".
[4] Benjamin Zhu, Kai Li, Hugo Patterson, "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System".
[5] Google, MSDN, Stackoverflow.