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 4, April 2017

A Secure Data Deduplication Scheme for Cloud Storage System

Bhos Komal[1], Ingale Karuna[1], Hattikatti Susmita[1], Jadhav Sachin[1], S.S. Mirajkar[1], S.R. Kakade[1]

Department of Computer Engg, JSPM’s RSCOE, Tathwade, Pune, Maharashtra[1]

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. There is only one copy for each file stored in cloud even if such a file is owned by the number of users. As a result, 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. It is Aim to address the above security challenges; this paper makes the first attempt to formalize the notion of distributed reliable system. We propose new distributed 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 systems. Security analysis demonstrates that our 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: Cloud storage; deduplication; delaydedupe; replica.

I. INTRODUCTION

Cloud storage is a cloud computing model in which data is stored on remote servers accessed from the internet, or cloud. The rapid growth of cloud storage, users and enterprise would like to take the backup of their data on cloud storage. The storage pressure on cloud storage system is growing by the day. IDC found that nearly 75% of data in the information systems is redundant. The data deduplication technique can identify the redundant data and eliminate duplicated data.
Data deduplication is an efficient data reduction technique, that can stores only one copy of each data by comparing their fingerprints with the existing stored data to identify duplicated data. This process of deduplication split into four stages 1. Chunking 2.Calculating fingerprint 3.Fingerprint lookup 4.Storing new data. Chunking can break the incoming large data into small portion called the ‘chunks’. Calculating fingerprint hash algorithm MD5 or SHA-1 is used after this unique identifier value is assigned to every chunk. In Fingerprint lookup the new incoming data are compared with the already stored data using the unique identifier. If the same identifier found, the redundant chunk will be removed and update the reference point otherwise, a new chunk will be stored. A new approach combining client-side deduplication and target-side deduplication for cloud storage system. The contribution of this is twofold. Firstly, file-level and chunk-level duplication are detected and eliminated locally by Client and globally by Metadata Server (MS) to improve the deduplication ratio. Secondly, we put forward the DelayDedupe strategy a delayed target-deduplication scheme based the chunk level deduplication and the access frequency of chunks in the Snodes. Combined with replica management, this method considers the value of duplicated data itself and determines whether new duplication for data modification is hot by the access frequency.

II.RELATED WORK

Data uploading, such as Cumulus [20] and Dropbox, in order to reduce network bandwidth, but takes up a large number of computing resources at the source end. Compared with client-side deduplication, server-side deduplication (also known as target deduplication) happens at the target end where data is stored so it can eliminate redundancy between different data sources to ensure that only one copy is stored. Inline deduplication can realize real-time data reduction when data is written to the storage system, such as Data Domain, but consumes much system resources.
Offline deduplication requires enough available free space for data storage before deduplication, and this approach is suitable for the storage protocols like Direct-Attached Storage and Storage Area Network. File-level deduplication, chunk-level deduplication and byte-level deduplication can increase storage utilization by eliminating data redundancy within or across files. Whole-file chunking cannot eliminate data redundancy within files, so finer-grained chunking strategies like fixedsize chunking, content-defined chunking, and TTTD chunking are introduced. In fixed-size chunking algorithm, all files are portioned into blocks with a fixed size such as 4KB and 8KB. This approach is suitable for EXE, PDF, DOC, TXT files due to its low computational overhead.

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 4, April 2017

Nevertheless, it is very sensitive to the change of data, for example, the chunking boundaries are offset after data insertion (even a byte of data). Different from fixed-size chunking, content-defined chunking as a typical variable size chunking algorithm combines Rabin's fingerprint [21] and sliding window to determine the chunking boundaries by the content of data, and the similarity of files can thus be detected. Found when the window size is 48 bytes and the expected chunk size is 8KB, content-defined chunking can provide better performance of deduplication.
However, it will result in high system overhead and extreme cases that the size of data chunk is too big or too small. Therefore, the improved content-defined chunking algorithm effectively controls the distribution of block size by setting the minimum and maximum chunk size.

After all files are divided into small blocks, the fingerprints of all blocks need to be calculated as their identifiers by using MD5 or SHA-1 and the identifiers are used for fingerprint lookup. Since the probability of hash collisions are orders of magnitude smaller than probability of disk corruption they are almost ignored by the existing research results. That is, two fingerprints are the same if and only if their corresponding two data are the same. Moreover, although both cryptographic hash functions take as input a message of arbitrary length, they produce as output a specific bit length of fingerprint, 128-bit for MD5 and 160-bit for SHA-1, respectively. In our system we provide the security framework using AES algorithm. AES is Advanced Encryption Standard. It is symmetric block cipher. It uses same key for encryption and decryption.

III.SYSTEM ARCHITECTURE

Our proposed system is the security mechanism; means provide the security to the MS i.e, metadata server. It may be happen to hack the users important data for that purpose AES algorithm is used to secure the users data on metadata server. AES algorithm is the more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays. AES stands for Advance encryption standard. It is found at least sixs times faster than triple DES. AES performs all its computations on bytes rather than bits.

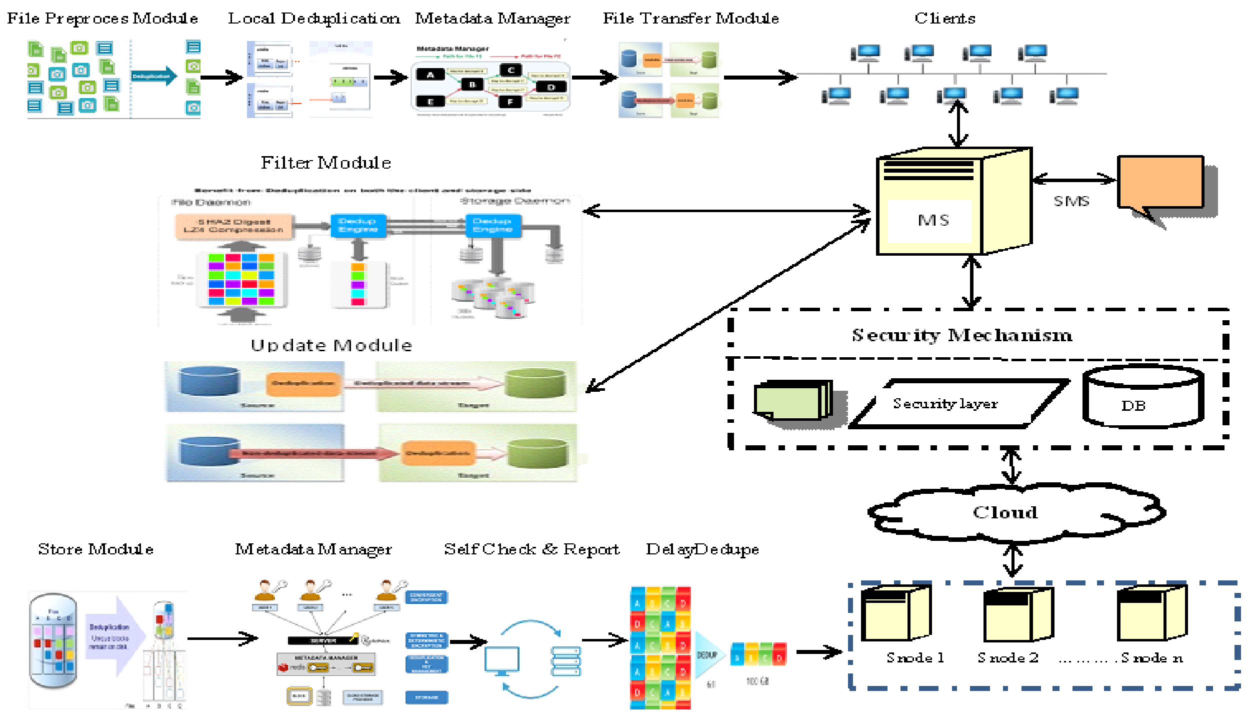

Fig no.01

As shown in Fig. 1, the system has Client, MS, Secondary MS (SMS) and Snode. This represent the location of the original data to be uploaded and the location of the new data to be stored after deduplication. User sends request of file uploading, access, modification, and deletion through Client. All the metadata information of files are stored in MS and the original data into Snode. Using MS information, we find location of data in Snode and check whether the data from Client is duplicated. MS plays very main role as the core manager of the whole system architecture. If MS fail to function, then the whole system will not work properly. To avoid this failure, SMS is used for synchronizing the backup of metadata images and operation logs. Each time we upload the some data from Client to Snode, firstly we

deal with these information by local deduplication and upload their metadata information to MS. Then we wait for the answer about non-duplicated data from MS, and finally upload the new data to Snode. There are four modules on the Client side**:** File process Module, Local Deduplication**,** data Manager, and File Transferred Module. The file process module is used to classify the files into dissimilar groups by using their types and calculates the hash value of files by using MD5 algorithm. Local Deduplication**:** It is responsible for eliminates the duplicate copies through file level and block level deduplication. Metadata Manager: It manages the fingerprints of files and chunks that uploaded to Snode to prevent the duplicated data from being uploaded again and again. File Transfer Module: This module used totransfer the data to MS, then metadata of these information processed by localdeduplication and it will not upload the new data until the "not found" response of MS is received.MS has two modules: Filter Module and Update Module. The filter module used to filtering duplicated data from disparate clients. The Update Module used to Updates the metadata index in MS to modify the metadata information from Snode. Snode **:** The storage node has four modules on the Snode side: Store Module, Metadata Manager, Self-check &Report Module, DelayDedupe Module. Store Module: This module used to store the original data blocks on the disk.
Metadata Manager**:** It manages the metadata information including fingerprint and reference of the chunk stored in Snode. Self-check & Report Module: It detects the duplicated data for data modification by other users and reports this modified information to MS. DelayDedupe Module: It checks whether the duplicated chunk is hot.

IV.TECHNIQUES AND ALGORITHMS

A. Cloud Computing:-

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly.
Cloud computing is a practical approach to experience direct cost benefits and it has the potential to transform a data center from a capital-intensive set up to a variable priced environment.
The idea of cloud computing is based on a very fundamental principal of „reusability of IT capabilities'. The difference that cloud computing brings compared to traditional concepts of "grid computing", "distributed computing", "utility computing", or "autonomic computing" is to broaden horizons across organizational boundaries.
Cloud storage systems are able to provide low-cost and convenient network storage service for users, which makes them more and more popular. However, thestorage pressure on cloud storage system caused by the explosive growth of data is growing by the day, especially a vast amount of redundant data waste plenty of storage space. Data deduplication can effectively reduce the size of data by eliminating redundantdata in storage systems.

B.MD5 Algorithm:-

The MD5 message digest algorithm is simple to implement and provides "fingerprints" .It takes as a input a message of arbitrary length and produces as output a 128 bit "fingerprint" or message digest of the input.

There are 5 steps:
Step 1 – Append padded bits:
The message is padded so that its length is congruent to 448, modulo 512. Means extended to just 64 bits shy of being of 512 bits long. A single "one" bit is appended to the message, and then "0" bits are appended so that the length in bits equals 448 modulo 512.

Step 2 – Append length:
A 64 bit representation of b is appended to the result of the previous step. The resulting message has a length that is an exact multiple of 512 bits.

Step 3 - Initialize MD Buffer:
four-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D, is a 32 bit register. The register are initialized to the following values in hexadecimal

A
Word A: 01 23 45 67
Word B: 89 ab cd ef
Word C: fe dc ba 98
Word D: 76 54 32 10

Step 4 – Process message in 16- word blocks:
Four auxiliary that take as input three 32-bit words and produces as output one 32-bit word.

F(X, Y, Z) = XY ∨ not(X) Z
G(X, Y, Z) =XZ ∨ Y not (Z)
H(X, Y, Z) =X xor Y xor Z
I(X, Y, Z) =Y xor (X ∨ not (Z))
If the bits of X, Y and Z are independent and unbiased the each bit of F(X, Y, Z), G(X, Y, Z), H(X, Y, Z), and I(X, Y, Z), will be independent and unbiased.

Step 5 – Output
The message digest produced as output is A, B, C, D. That is, output begins with the low-order byte of A, and end with the **higher order byte of D.**

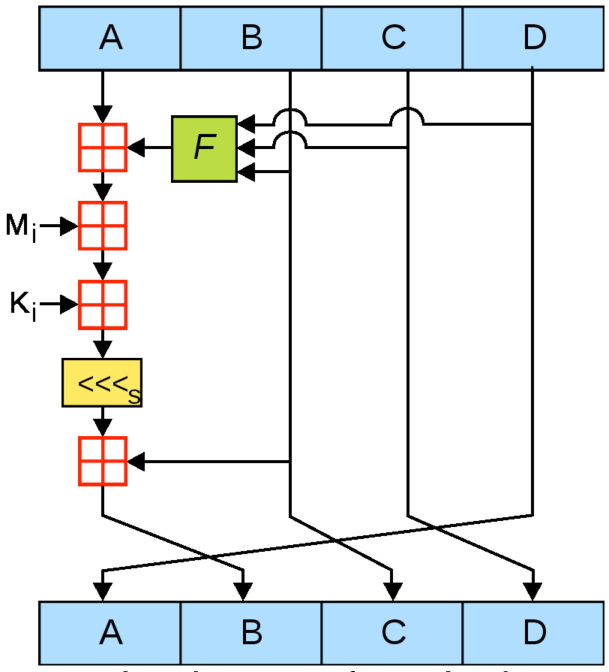

Fig-shows the operation of MD-5 algorithm

C.SHA-1 Algorithm:-
Step 1: Append Padding Bits….
Message is “padded” with a 1 and as many 0’s as necessary to bring the message length to 64 bits fewer than an even multiple of 512.

Step 2: Append Length....
64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

Step 3: Prepare Processing Functions….
SHA1 requires 80 processing functions defined
f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)
(0 <= t <= 19)
f(t;B,C,D) = B XOR C XOR D
(20 <= t <= 39)
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <=59)
f(t;B,C,D) = B XOR C XOR D(60 <= t <= 79)

Step 4: Prepare Processing Constants....
SHA1 requires 80 processing constant words defined as:
K(t) = 0x5A827999
(0 <= t <= 19)
K(t) = 0x6ED9EBA1
(20 <= t <= 39)

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 4, April 2017

K(t) = 0x8F1BBCDC
(40 <= t <= 59)
K(t) = 0xCA62C1D6
(60 <= t <= 79)

Step 5: Initialize Buffers….
SHA1 requires 160 bits or 5 buffers of words (32 bits):
H0 = 0x67452301
H1 = 0xEFCDAB89
H2 = 0x98BADCFE
H3 = 0x10325476
H4 = 0xC3D2E1F0

Step 6: Processing Message in 512-bit blocks (L blocks in total message)….
This is the main task of SHA1 algorithm which loops through the padded and appended message in 512-bit blocks.
Input and predefined functions:
M[1, 2, ..., L]: Blocks of the padded and appended message f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): 80 Processing Functions K(0), K(1), ..., K(79): 80 Processing Constant Words
H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values

Step 7: Pseudo Code….
```
For loop on k = 1 to L
(W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
For t = 16 to 79
W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1
 A = H0, B = H1, C = H2, D = H3, E = H4
 For t = 0 to 79 do:
TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t) E = D, D = C,
C = B<<<30, B = A, A = TEMP
 End of for loop
 H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D,
H4 = H4 + E
              End of for loop
```
Step 8:Output:
H0, H1, H2, H3, H4, H5: Word buffers with final message digest

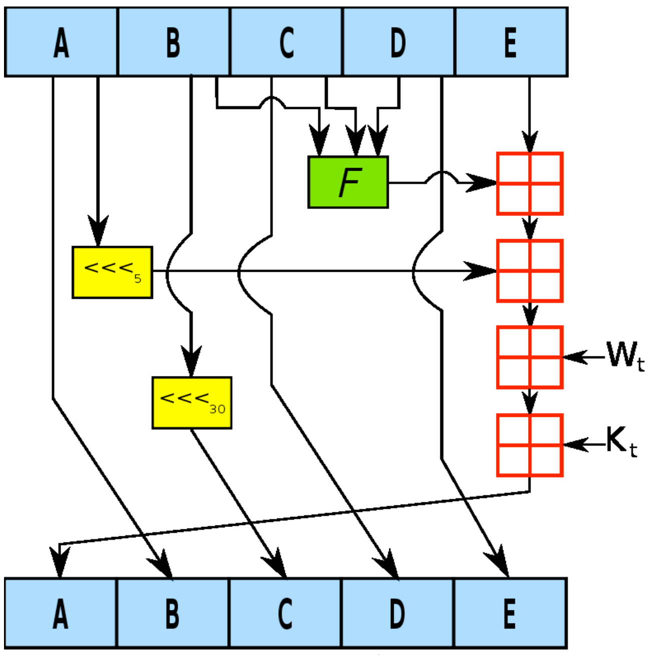

Fig-shows the operation of SHA-1 algorithm

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 4, April 2017

D.AES Algorithm:-

AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. However, AES is quite different from DES in a number of ways.
The algorithm Rijndael allows for a variety of block and key sizes and not just the 64 and 56 bits of DES' block and key size. The block and key can in fact chosen independently from 128, 160, 192, 224, 256 bits and need not be the same. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits.
Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES-256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure. Recall that in a feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. In this case the entire data block is processed in parallel during each round using substitutions and permutations.
A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key.

Inner Working of Round:-

The algorithm begins with an **Add round key** stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm. The four stages are as follows:
1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the **Mix Columns** stage. The first nine rounds of the decryption algorithm consist of the following:
1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

Again, the tenth round simply leaves out the **Inverse Mix Columns** stage.

Step1:

Called SubBytes for byte-by-byte substitution during the forward process. The corresponding substitution step used during decryption is called InvSubBytes.

• This step consists of using a 16 × 16 lookup table to find a replacement byte for a given byte in the input state array.
• The entries in the lookup table are created by using the notions of multiplicative inverses in GF(28) and bit scrambling to destroy the bit-level correlations inside each byte.

Step2:

called ShiftRows for shifting the rows of the state array during the forward process. The corresponding transformation during decryption is denoted InvShiftRows for Inverse Shift-Row Transformation.)

• The goal of this transformation is to scramble the byte order inside each 128-bit block.

Step3:

Called MixColumns for mixing up of the bytes in each column separately during the forward process) (The correspond-ingtransformation during decryption is denoted InvMixColumns and stands for inverse mix column transformation.) The goal is here is to further scramble up the 128-bit input block.

• The shift-rows step along with the mix-column step causes each bit of the ciphertext to depend on every bit of the plain-text after 10 rounds of processing.

Step 4: called AddRoundKey for adding the round key to the output of the previous step during the forward process. The cor-responding step during decryption is denoted InvAddRound-Key for inverse add round key transformation.

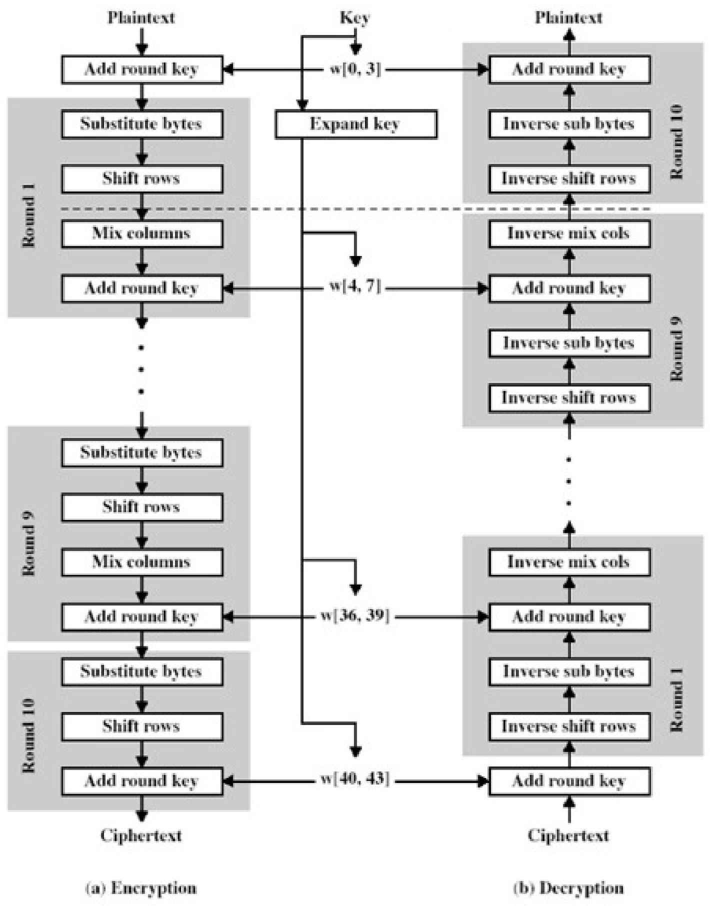

fig-shows the working of AES algorithm.

V. DELAYED DEDUPLICATION

Data deduplication is a technique whose objective is to improve storage efficiency. With the aim to reduce storage space, in traditional deduplication systems, duplicated data chunks identify and store only onereplica of the data in storage. Logical pointers are created for other copies instead of storing redundant data. Deduplication can reduce both storage space and network bandwidth. However such techniques can result with a negative impact on system fault tolerance. Because there are many files that refer to the same data chunk if it becomes unavailable due to failure can result in reduced reliability. Due to this problem, many approaches and techniques have been proposed that not only provide solutions to achieve storage efficiency, but also to improve it fault tolerance. These techniques provide redundancy of data chunks after performing deduplication.

When data access is overheating, increasing the replica copies can usually reduce the system bottlenecks and improve the availability of data. Meanwhile, if data is rarely accessed, too much replicas will increase the system cost including the cost of storing and updating replicas .Different replicas of the same data must be synchronous and updated to ensure that the reference count and user authority of chunks are identical in different S nodes. But the access num of chunks depends on the performance of S nodes.

VI.RESULT AND ANALYSIS

The authorized deduplication system used to avoid duplicate copies of data in the given cloud. Proposed system implemented by using block level duplication which compare the given blocks with database, suppose the file is already stored in the database and that same file uploaded by another user at that time only metadata of file will be store not actually file so it reduce the storage space of data and proper utilization of space. The data will be store in encrypted format so it also maintains security because each block contains their own token, cipher text and private key. The database size will be reduced by using this technique. The proposed system has been compared with the existing system on the basis of database usage, and security using AES algorithm.

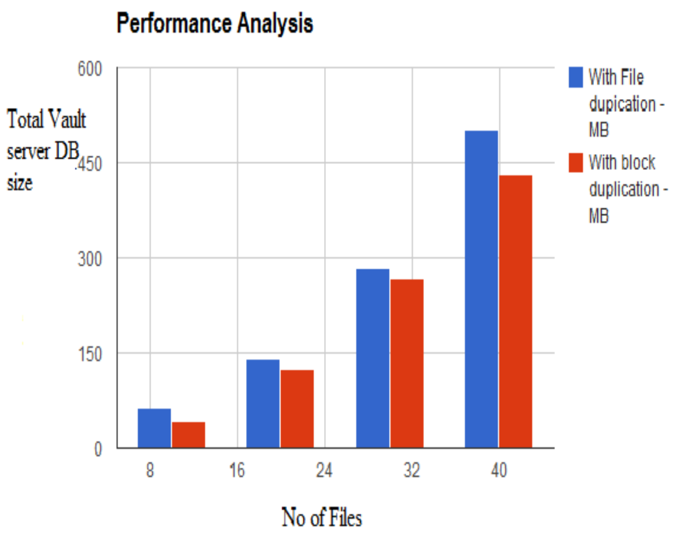

Fig:-shows the comparison between block level and file level Deduplication.

The above Fig..show expected output from proposed system. X axis shows number of files and Y axis shows total database size. It shows actual storage space in database, with file level duplication having large storage space as compare to block level duplication. For that purpose, use the block level duplication for reduce storage space on cloud and reduce duplication.

VII.CONCLUSION

We proposed the distributed de-duplication 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 de-duplication. The security of tag consistency and integrity were achieved. We implemented our de-duplication systems using the Ramp secret sharing scheme and demonstrated that it incurs small encoding/decoding overhead compared to the network transmission overhead in regularupload/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 digital 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 de-duplication,” 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. . 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 de-duplication with efficient andreliable convergent key management,” in IEEE Transactions on Parallel and Distributed Systems, 2014, pp. vol. 25(6), pp. 1615–1625.
[12] N. Kaaniche, and M. Laurent, ”A secure Client side deduplication scheme in cloud storage environments,” in Proc. 20146th Int. Conf. New Technol., Mobility and Security (NTMS), Dubai, 2014, pp. 1-7.