SECURE DATA AUDITING WITH PROXY CODE REGENERATION IN CLOUD DATA DEDUPLICATION ENVIRONMENT

Miss. Arti L. Bairagi[1],
Prof. P. M. Yawalkar[2]

[1] *M.E. Student, Department of Computer Engineering, Savitribai Phule University, Nasik, India*
[2] *Assistant professor, Department of Computer Engineering, Savitribai Phule University, Nasik, India*

ABSTRACT

In last few decades cloud computing technology is widely developed. At the user's end, outsourcing of data becoming an attractive trend to release heavy management of data. There are certain problems have been introduced during existing system analysis. Among these entire problem we proposed an approach which gives the relief for the problem of data duplication on cloud server as well as integrity check of original data. Cloud & users unaware of the type of data that previously stored on cloud server & this is the main reason behind data deduplication check. In our system there are two types of secure modules have been used namely, SecCloud & SecCloud+. Data tags are generated at the time of data uploading on cloud for duplication check. For secured data deduplication on encrypted data SecCloud+ is implemented. Apart from this work system provide the data recovery by contributing new data recovery concept known as, "erasure coding". With experimental setup our system works efficient in terms of storage and security parameters. It reduces the cloud server bandwidth & also saves the storage space.

Keyword: - *Proof of retrievability, proof of ownership, deduplication, cloud storage*

1. INTRODUCTION

Cloud storage is the model enterprise network in which data is stored in virtualized pools of storage which is hosted by third parties. There are different types of facilities provided by the cloud such as, less expensive services, simplified convenience etc. From the study of existing system, it is aimed that 40 trillion GB data is expected to store & utilized on cloud server [4]. Many services provided by the cloud to the third parties which have some limitations such as, data duplication, integrity verification, problem during data transmission etc. There are main two problems in cloud computing such as, data deduplication and data integrity check due to rapid growth of outsourcing data to the cloud. As per the analysis of J. Li, 75% digital data is duplicated copies. Data deduplication introduced a new technology namely, data deduplication which means there is need of reduction of duplicate data from cloud server. De-duplication leads to remove important threats from the storage system [2][3]. In proposed deduplication approach cloud notify to their users or clients if similar data is available on it. This approach is static which causes another problem of data integrity on cloud server. There are many cases in which data integrity check problem is visualized. Proposed system works on the problem of data deduplication & data integrity check. SecCloud and SecCloud+ have been represented in this work. SecCloud is the comprehensive language based approach implemented for data integrity auditing similar to MapReduce cloud. It suggest user to create tag before uploading it on cloud. It also provides fined-grained functionality & also enables secured de-duplication approach in cloud storage. It mainly works to preserves the leakages of side channel information. The user generated tags are preserved as hash value at the auditor's end. Proof of relationship is designed between cloud server & end user's. It verifies that the client owns claimed file exactly. PoW is conducted at the time of file uploading procedure. Cloud server works as authenticator in

PoW. Another is SecCloud+ which is proposed by motivating from the concept of data encryption which gives the assurance of data confidentiality. Convergent encryption is utilized to directly audit data integrity on encrypted data. Data deduplication is the prevention of dictionary attack. To protect convergent key form deriving it from contents of file some modifications is made in convergent encryption technique such as, convergent key of file is produced and restrained by a secret "seed". As per above discussion, while performing task of data deduplication check there may have chances of data leak hence, this system contributes proxy server to preserve regenerating data which is generated at cloud server's end after data is uploaded by user or third parties. It can be efficient way of data recovery in cloud storage[1].

2. RELATED WORK

G. Ateniese, et al. [2] [3], suggested two PDP approaches which is provable secure. In this paper, PDP scheme is extent by I/O rather than the cryptographic estimations. I/O establishes data consumptions inference with on-demand bandwidth to store and retrieve data. Proposed technique remarked needs of user's which enables authentication of server maintained file data without fetching it from server. They analyze issues of existing techniques such as, complexity of storage. In this system PDP model is used to give third party file storage. Here user can store feasible amount of metadata which can be used to verify the proof of server. It uses the bandwidth of O (1).

This discussion about detection of data insufficiency corruption is continued by author in [3]. Problem of integrity auditing is their main focused for which they have represented PDP model is to minimize desirable accesses to file blocks. Some solutions have been proposed which can obtain some overhead at the server's end which contains some fixed amount of communication amount per challenge. Spot checking is introduced to ensure that the proposed scheme handles lightweight, verifiable homomorphic tags for authenticating data occupation without accessing original data file. A remote data checking technique known as, RDC is implemented for robust auditing. In this process, it integrates RDC with FEC i.e. forward error correcting codes. It helps to reduce fewer data corruption.

G. Ateniese, L.V. Mancini [4], represented a scalable and efficient PDP technique which is based on symmetric key cryptography. The proposed PDP does not required bulk encryption. It allows outsourcing of dynamic data. Its primary focused is to allow user to authenticate the server in efficient and secure way to store large amount of data. It can also utilize for untrusted and malicious server. Proposed technique required number (t) of verification to be preassign. Limited storage space is required to obtain data. For number of blocks there is requirement of independent numbers of tokens. From overall analysis, there are some salient features have been identified such as, outsourcing of dynamic data, minimum cost.

H. Shacham et al [5], Q. Wang et al. [6], J. Xu et al.[8], provides the solution for the ideal challenge such as, efficient and demonstrable data security. All they have proposed POR technique. It is abbreviated form of proof of retrievability. It is implemented to provide security proofs against the arbitrary adversaries in the powerful model. It builds BLS signature and a secure random oracle model. It mainly aims to provide data verifiability. Pseudo Random Functions (PRF) technique is proposed to provide private as well as secured verification of data. To break files into 'n'- number of blocks erasure code technique have been proposed in this paper. They proposed Reed-Solomon codes for large file encoding and decoding but it required more time processing. Two homomorphic authenticators are discussed namely, PRF and BLS signature. PRF technique is proposed for proof of retrievability whereas; BLS signature is used for public verifiability.

Q. Wang, C.Wang et al. [6], discussed issues of data integrity. For integrity verification third party auditor (TPA) is represented which verifies the data on behalf of user. TPA can also verify dynamic data such as, operations, insert, delete, update etc. Dynamic data verification is the main advantage of POR. It can also deduct the user's interference. Using POR blockless verification is also achieved. Merkle hash tree (MR-MHT) is constructed

to extend POR model. For dynamic data verification fixed size of communication bits per verification and 1/s storage overheads occurred [7].

E. Stefanov, et al., [8], discussed about Iris system. It is authenticated file system. It is designed to support workload of storing data from large organization. Strong data integrity, freshness of data and efficient data retrievability is provided by Iris. It is scalable approach. It is suitable for outsourcing of files of large organizations or enterprises. MR-MHT is constructed for authentication of file system. To overcome an economical barriers in transmigrate cloud storage Iris is the best solution. MR-MHT is more complex component of Iris. It can achieve degree of end-to-end optimization via crafted and holistic architecture. From analysis of system, some overheads have been observed with Iris such as, a common bottleneck to the security system deployment which affects on system performance.

Jin Li, X. Tan et al [9], proposed architecture of cloud storage with two independent servers. From mentioned two servers one is cloud server and the other is cloud audit server. Audit server is mainly to eliminate the user's involvement in the phases of auditing and pre-processing. There are many challenging issues in cloud such as, scalability, security and performance of complete system. In this paper author discussed about RSA-based homomorphic tags for auditing outsourced data.RSA aggregates the file blocks into single block and verifies the homomorphic property of RSA. It is useful when data owner computes the large number of tags for outsourced data which involves the data exponential and multiplication operations.

S. Halevi, D. Harnik [10], identified attacks which stroke client-side deduplication. It allows attacker to gain access to irrational file size of other user based on very small hash signatures of these files. In this paper, the technique POW's is proposed to overcome such attacks. PoW is Proof of Ownership. Under the rigorous security definition the concept of PoW is formalized. They have represented Merkel hash tree construction and specific encoding techniques for efficient implementation of PoW. The proposed technique is somewhat similar to the POR. In this paper, for meaningful security author optimized more practical solution. There are three definitions, three matching protocols and security settings given in this paper. The proposed protocol allows setting the threshold to minimum summary file. Limitation of the proposed scheme is it cannot protect CDN attacks.

R.D. Pietro, A. Sorniotti[11], introduced PoW. It has all features like state-of-art solution. Another proposed mechanism is information theoretic rather than computational complexities. PoW is designed to allow server to verify whether the user's owns file. The proposed scheme PoW attains several goals such as, the cost of I/O computations not depend on input, another is the proposed approach is efficient for wide range of systems parameters and it is information- theoretically secure. There two phases have been given in the proposed PoW scheme, in first phase, cloud receives the file and pre-computes the responses for number of PoW challenges which relates to file. The second phase is triggered by the client when it sends to the server a unique identifier for a file it wishes to prove possession of.

W.K. Ng, Y. Wen et al [12], proposed a new notion private data deduplication protocol. It is deduplication technique used for private data storage. A development of private de-duplication protocol is based on some standard cryptography assumptions. They represented that the proposed technique is secure assuming that the crucial hash function is collision- resilient, the discrete logarithm is difficult and erasure coding algorithm can erasure up to specific fractions of the bits in the presence of malicious adversaries in the presence of malicious adversaries.

J.R. Douceur, A. Adya [13], proposed a mechanism which contains convergent key encryption and SLAD i.e. self-Arranging, Lossy, Associative Database for integrating content file as well as information of location in decentralized, scalable, fault-tolerant way. The proposed technique identifies the identical files in distributed file system. Author developed convergent encryption cryptosystem to produce identical plaintext files regardless of their encryption keys. For convergent key encryption user first computes strong cryptographic hash of the file

content. Then file encryption is performing using generated value as key. Then the problem of performing files identification across large number of desktop machines is solved by using SALAD. SALAD stores the file location and content information in distributed way.

3. SYSTEM ARCHITECTURE

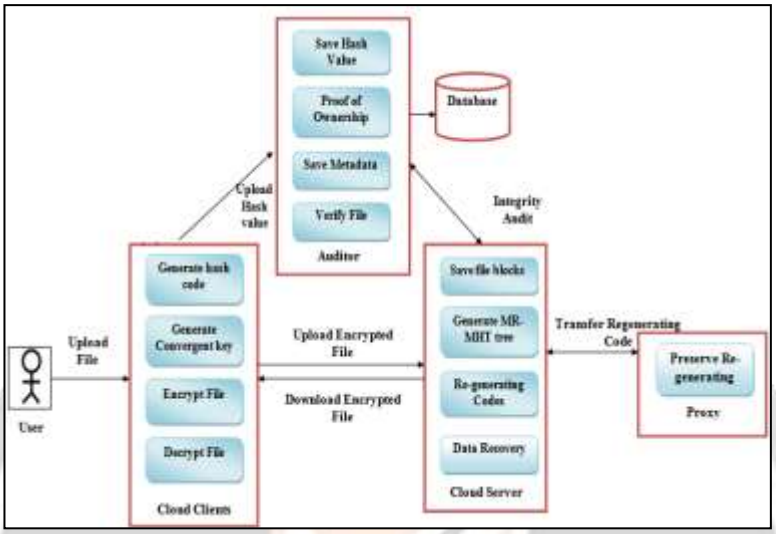

Fig.1 System architecture diagram

Above figure 1 represents the proposed system architecture. Proposed system aims to achieve data integrity and deduplication in cloud, the two secure systems is namely SecCloud and SecCloud+. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud.

Besides supporting integrity auditing and secure deduplication, SecCloud+ enables the guarantee of file confidentiality. The method of directly auditing integrity on encrypted data is given. The different modules of system architecture are as follows:

Before interacting with system, user has to register and then login to system. Then rest procedure to interacting with each module of the system is given as follow:

1. **Cloud clients:**
   Cloud Clients have large data files to be stored and rely on the cloud for data maintenance and computation. They can be either individual consumers or commercial organizations

2. **Cloud Server:**
   Cloud Servers virtualized the resources according to the requirements of clients and expose them as storage pools. Typically, the cloud clients may buy or lease storage capacity from cloud servers, and store their individual data in these bought or rented spaces for future utilization.

3. **Auditor:**
   Auditor helps clients upload and audit their outsourced data maintains a MapReduce cloud and acts like a certificate authority. This assumption presumes that the auditor is associated with a pair of public and private keys. Its public key is made available to the other entities in the system.

4. **Proxy:**
   At the end of proxy it will regenerate the data as per the auditor request and preserve it.

4. ALGORITHMIC STRATEGY

1. AES Algorithm

1.1. AES Algorithm:

Input: Plain text message m in Byte [] , Key k
Output: Cipher text message in byte []
Processing steps:

1. Define 4 * 4 state array
2. Define constant Nr = 4, R=16
3. Copy m in state[]
4. Add each byte of state[] to key k using ⊕
5. For Nr-1 rounds Replace every byte in state[] with new value using lookup table Shift last 3 rows of state[] upside cyclically combine last 4 columns of state[] Add each byte of state[] to key k using ⊕ end For
6. Shift last 3 rows of state[] upside cyclically
7. Add each byte of state[] to key k using ⊕
8. copy State[] to output[]

1.2. AES Decryption:

Input:
Cipher text message C in byte[], Key k
Output: Plain text message m in Byte[] Processing steps:
1. Define 4 * 4 state array
2. Define constant Nr = 4, R=16 ,
3. Copy C in state[]
4. Add each byte of state[] to key k using L
5. For Nr-1 rounds Inverse Replace every byte in state[] with new value using lookup table Inverse Shift last 3 rows of state[] downside cyclically combine last 4 columns of state[]
Add each byte of state[] to key k using L end For 6. Inverse Shift last 3 rows of state[] down word cyclically
7. Inverse Add each byte of state[] to key k using L
8. copy State[] to output[]

2. Merkel Hash Tree Algorithm

A.Algorithm for Data update and verification:
A:Notation
F:Uploaded file
m_i: i^{th} block of file
$b_{i,j}$:The i^{th} block of replica F_j
T: Replica markel hash tree developed based on $\{m_i\}$
T_i: Replica-sub tree of block $\{m_i\}$
+'+Ω_i:A set of tuples that are used as, m_i +' +s auxiliary authentication information(AAI)
R:value stored in root of tree,this is hash value
$^{\delta}{}_{I,j}$: The homomorphic authenticator for $b_{i,j}$
$sign_{AUTH}$: Authorization signature for verification of TPA. Numbers, headers and footers must not be used.
I: insertion, M: update and D: deletion of data

The replicas of data required to be updated when the original data get updated. Also data verification required to be update. At the end of user, file block is generated and replicas are generated with respect to the type of request - I/M or D. Communication between client and cloud while updating the data is given in following algorithmic steps.

Algorithm:

Step 1: Client upload file with its block $m_{i,j}$ for new upload (I)/modifications (M)/deletion(D)
Step2: Cloud compute $b_{i,j}$ based on m_i then generate update notification as, $\{M/1i, \{b_{i,j}\}\}$ Or $\{D,I\}$ $\{M/1_i,\{b_{i,j}\}\}$ Or $\{D_i\}$

Step 3: CSS locate subtree Ti, and compute R with $\{b_{i,j}, \Omega_i\}$
For I/M: create new sub tree with $b_{i,j}$ and update following indices:
$\{h(b_{i,j}), +' + \Omega_i, R +' +, sign\}$
For D: delete Ti and updated indices and then compute R +' +
Step4: Cloud compute +' + σ =(R, u, $b_{i,j}$) and sig+'+=(H(R+'+)) and send to TPA as metadata.

A. Algorithm for data verification:

Original data blocks & replica present on CSS are used for data verification. In this TPA sends the challenge message to CSS and CSS creates the proof for the received request. It computes σ and ψ for each single block and its replica then sends response to the TPA. TPA verifies the generated proof and generates a response as accept or reject. This response is conveyed to the user as verification result. Following algorithm shows the communication between TPA and CSS.

Algorithm:

Step 1: TPA Send Challenge request to CSS as:
$\{sign_{AUTH}, i, v_{i,j}\}$
Step 2: Css verify $sign_{AUTH}$
Then
Compute, $\mu_i = \Sigma_j v_{i,j} b_{i,j}$
$\sigma_j = \Pi_{i \in E} \sigma_{I,j} v_{i,j}$
$\{\mu_i, \sigma_j, \{H\{ b_{i,j} \}, h(b_{i,j}), \Omega, sig\}\}$ and send to TPA
Step 3: TPA compute & verify R and indices of σ_j with $\sigma +'_j$
Verify *sig* with R,
Then
Verify following *c* eq.: $e(\sigma j, g) = e(\Pi i\ H(bi,j)\mu, v)$
Step 4: TPA accepts, if all verifications passed else reject and generate audit report and send to the client.

3. SHA-1 algorithm

Input:
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0
ml = message length in bits (always a multiple of the number of bits in a character).

Processing:
1. Append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8 bits.
2. Append 0 to generate mod 512 value
3. Process the message in successive 512-bit chunks:
   for each chunk
   Break chunk into sixteen 32-bit big-endian words w[i]
4. Extend the sixteen 32-bit words into eighty 32-bit words:
   for i from 16 to 79
      w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

5. Initialize hash value for this chunk:
   a = h0,b = h1,c = h2,d = h3,e = h4
6. Apply Main loop with 80 rounds
      For 1st 19 round perform
        f = (b and c) or ((not b) and d)
        k = 0x5A827999
      For 20 to 39 round perform
        f = b xor c xor d

k = 0x6ED9EBA1
For 40 to 59 round perform
f = (b and c) or (b and d) or (c and d)
k = 0x8F1BBCDC
For 60 to 79 round perform
f = b xor c xor d
k = 0xCA62C1D6

7. Update hash value
temp = (a leftrotate 5) + f + e + k + w[i]
e = d
d = c
c = b leftrotate 30
b = a
a = temp

8. Add this chunk's hash to generate cumulative sum for h0 to h4 for a, b, c, d, e

9. Produce the final hash value (big-endian) as a 160-bit number:
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4
Output: Hash Value hh

4. Shamir Secret algorithm

Input: Secret 'k'

No. of parts 'n'

Shares 'k'

Processing steps:

Phase-I

Step 1: Define two random numbers

Step 2: Generate polynomial as, $f(x)=s+a_0^x + a_1^2$

Step 3: Construct 'n'-Points as

$Da_{x-1}=(x,f(x))$

Step 4: Generate 'n' shares and send

Step 5: Select 'k' shares i.e. $k(x_1, x_2, x_3)$

Phase-II

Step 1: Consider: k-shares or values i.e. $k(x_1, x_2, x_3)$

Such as, $(x_0, y_0)= x_1$

$(x_1, y_1)= x_2$

$(x_2, y_2)= x_3$

Step 2: Compute 'L' i.e. Lagrange Basis Polynomial.

Such as,

$$L1 = \frac{x - x_0}{x_1 - x_0} . \frac{x - x_2}{x_1 - x_2}$$

$$L2 = \frac{x - x_0}{x_2 - x_0} . \frac{x - x_1}{x_2 - x_1}$$

Therefore,

$$f(x) = \sum_{j=0}^{2} y_j . l_j(x) = a_0 + a_1 + a_2$$

Output:

a_0 i.e. secret 'S'

1. HMAC algorithm
Input: Key, message

Processing steps:

1. If (length(key)> blocksize) then
   Key =hash(key)
   End if
2. If(length(key)<blocksize) then
   Key= key // [0x00 * (blocksize-length(key))]
   End if
3. O_key_{pad} =[0x5c * blocksize] $\oplus$ key
   i_key_{pad} =[0x36 * blocksize] $\oplus$key
4. hh = hash O_key_{pad} // hash(i_key_{pad}//message)

Output: Hash Value hh

5. MATHEMATICAL MODEL

System S can be defined as:

S = {CC, CS, A, KS, P}

Where,

1. CC = {CCI, CCF ,CCO } is A Cloud Client

CCI = {CCI1, CCI2, CCI3, CCI4,	CCI1 = User Registration Details

CCI5}, A set of Cloud Client Input From User	CCI2 = User Login Details
	CCI3 = User File
	CCI4 = Destination Folder
	CCI5=Audit Request
CCF = {CCF1, CCF2, CCF3, CCF4, CCF5, CCF6, CCF7, CCF8, CCF9,CCF10,CCF11,CCF12, CCF13 }, A set Of Function	CCF1 = User Registration
	CCF2 = User Login
	CCF3 = Generate File Blocks
	CCF4 = Generate Hash Value
	CCF5 = Upload Hash Value
	CCF6 = Generate Convergent Keys
	CCF7 = Upload Key To KDS Server
	CCF8 = Encrypt File Blocks
	CCF9 = Upload File Blocks To Cloud Server
	CCF10 = Call Auditor For Data Auditing
	CCF11 = Download File Blocks
	CCF12 = Download Keys
	CCF13 = Decrypt File Blocks
	CCF14 = Save File
CCO ={CCO1,CCO2,CCO3}, A set Of Output	CCO1 = Success Note
	CCO2 = Downloaded file
	CCO3 = Audit Report

2. CS = {CSI, CSF, CSO} is A cloud Server

CSI = {CSI1, CSI2, CSI3, CSI4}, A set of Input	CSI1 = Encrypted File
	CSI2 = Download Request
	CSI3 = Challenge Message
	CSI4 = Regenerating Codes

CSF = { CSF1, CSF2, CSF3, CSF4, CSF5, CSF6, CSF7}, A set Of Function	CSF1 = Save Blocks
	CSF2 = Generate Merkel Hash Tree
	CSF3 = Generate Metadata
	CSF4 = Generate regenerating codes
	CSF5 = Generate Proof
	CSF6 = Download File
	CSF7 = Generate Data from Codes
CSO ={ CSO1, CSO2, CSO3, CSO4, CSO5 }, A set Of Output	CSO1 = Success/ Failure Note
	CSO2 = Metadata
	CSO3 = Regenerating codes
	CSO4 = Blocks For Downloading
	CSO5 = Proof

3. A = {AI, AF ,AO } is A Cloud Service Auditor

AI = {AI1,AI2,AI3,AI4}, A set of Input	AI1 = User Details
	AI2 = User File Metadata
	AI3 = Verification Request
	AI4 = Proof Of files
AF = {AG1, AF2,AF3,AF4,AF5}, A set Of Function	AF1 = Save User Details
	AF2 = Save Metadata
	AF3 = Generate Challenge Message
	AF4 = Verify Proof
	AF5 = Generate Audit Report
	AF6 = Generate regeneration Request
AO ={AO1,AO2,AO3}, A set Of Output	AO1= Challenge Message
	AO2 =Verification Result
	AO3 = regeneration request

4. PS = {PI, PF, PO } is A Proxy Server

PI = {PSI1,PSI2}, A set of Input	PSI1 = Regenerating Codes
	PSI2 = Regeneration Request
PF = {PSF1, PSF2}, A set Of Function	PSF1 = Save Regenerating Codes
	PSF2 = Get Regenerating odes
PO ={PSO1}, A set Of Output	PSO1 = Set Of Re-generating Codes

6. ANALYSIS OF RESULTS

System is divided into two sections, first section contains User side system and second section contains server side system as well as database. Server side system contains 3 parts: Cloud storage server, Auditor and proxy server. System GUI is designed for user interaction with desktop application and the database is configured with server side system.

Server side is designed using jdk-17 and tomcat whereas desktop application is designed using swing components. Mysql database is used at the server end.

Dataset used:

Synthetic dataset is generated for system testing. This dataset contains various text files with extensions like: xml, json, txt, java, .cs. For each type of file various files are collected with different sizes. The size varies from 2KB to 1000KB.

VI.RESULT TABLES AND DISCUSSION

TABLE I: FILE LEVEL DE-DUPLICATION CHECK

File size	**File level de-duplication check time**
200	1.95
400	2.02
600	2.07
800	2.11
1000	2.15

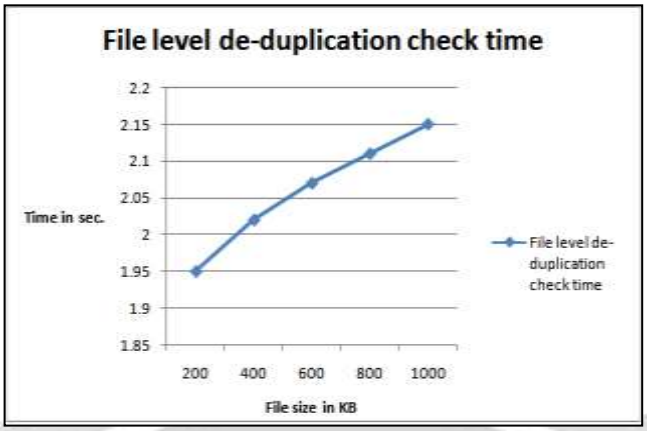

Figure 2: File level de-duplication check

In table I, time required for file level de-duplication check is given. For file level de-duplication we have used 200KB and 1000KB size of files. As per observation time required for 200KB file is 1.95sec. Whereas, time required for 1000KB file 2.15sec. There is very slight difference in each test file for file level de-duplication.

Figure 2 depicts graphical form of file level de-duplication in which X-axis contains file size in KB whereas Y-axis contains file de-duplication check in second.

TABLE II: BLOCK LEVEL DE-DUPLICATION CHECK

File size	**Block level de-duplication check time**
200	15.527
400	19.781
600	24.537
800	28.994
1000	32.367

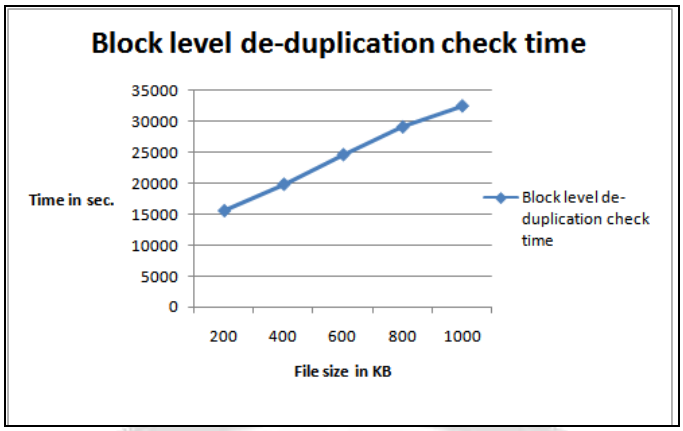

Figure 3: Graph block level de-duplication check

In table II, time required for block level de-duplication check is given. For block level de-duplication we have used 200KB and 1000KB size of files. As per observation time required for 200KB file is 15.527sec. Whereas, time required for 1000KB file 32.367sec. There is very slight difference in each test file for file level de-duplication.

Figure 3 depicts graphical form of file level de-duplication in which X-axis contains file size in KB whereas Y-axis contains file de-duplication check in second.

TABLE III: FILE VERIFICATION AND REGENERATION

File size	Verification time	Regeneration time
200	0.623	37.537
400	1.825	89.91
600	2.324	119.72
800	2.867	135.34
1000	3.079	159.948

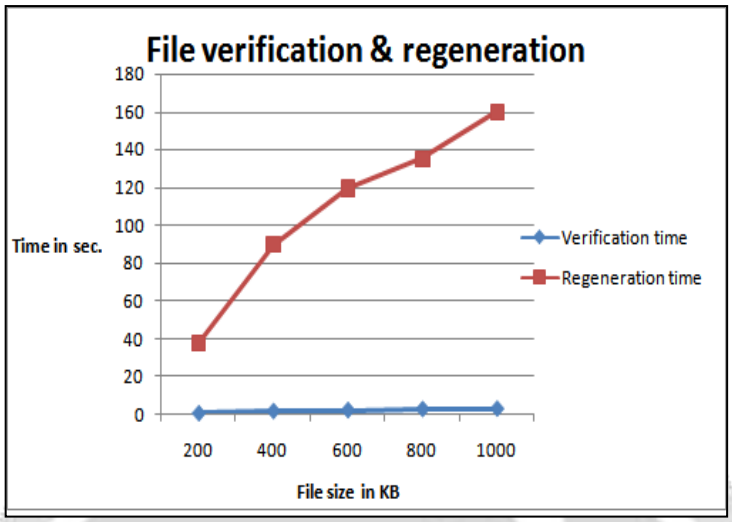

Figure 4: Graph file verification and code generation

In table III, time required for file verification and code regeneration is shown. For file verification and regeneration we have used 200KB and 1000KB size of files. As per observation time required for file verification is less than time required for regeneration time. Regeneration is performed only there exist any mismatch in cloud stored data.

Figure 4 depicts graphical form of file verification and code regeneration in which X-axis represents file size in KB and Y-axis represents time in sec.

TABLE IV: PERFORMANCE OF DE-DUPLICATION CHECK

File	25%	50%	75%	100%
200	1.965	3.93	7.86	15.72
400	2.497	4.995	9.99	19.98
600	3.093	6.186	12.372	24.744
800	3.65	7.301	14.602	29.205
1000	4.072	8.145	16.291	32.582

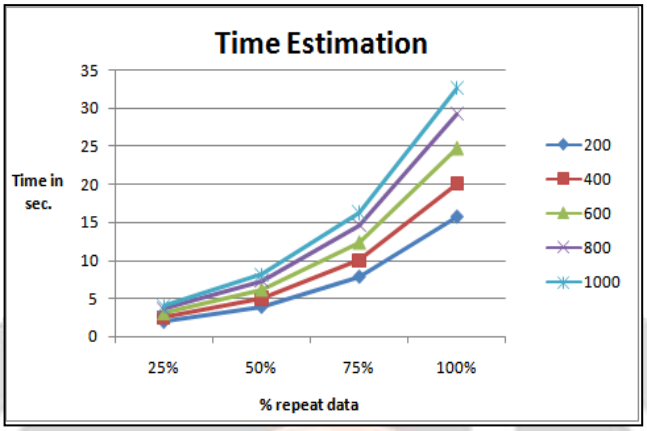

Figure 5: Graph time estimation

If file is partially present then only remaining blocks are uploaded to SecCloud and links are created for the file. We have tested this scenario for different cases where 25%, 50%, 75% file is already present on the server.

Above table IV shows the detailed description for partial file level de-duplication.

Paper	File level deduplication	Block level deduplication	Hash code/ Convergent Key	MHT-Tree	Sharing	Space efficient convergent key	Data auditing	Regene-ration
Block Level De-duplication Check	Y	Y	Y	N	N	N	N	N
Secure Distributed Deduplication	Y	Y	N	N	N	N	N	N
Secure Data Auditing with Proxy Code Regeneration	Y	Y	Y	Y	Y	Y	Y	Y

TABLE V: COMPARATIVE ANALYSIS

Table V gives comparative analysis between proposed and existing system. In proposed system, all features such as, file level de-duplication, block level de-duplication, convergent key generation, MHT-generation, data sharing, space efficient convergent key, data auditing and regeneration is provided. Therefore, proposed system is more secured and can efficiently work against duplication.

6. CONCLUSIONS

A secure data deduplication & data integrity auditing techniques has been proposed to reduce cloud space and bandwidth. Data transferred is takes place via internet from user to cloud or cloud to user vice-versa which may sometime causes data damage or leak hence data confidentiality get lost. SecCloud technique is proposed to address the problem of data integrity auditing whereas, advanced technique SecCloud+ can handles the process of data deduplication. PoW is designed between end user's\third parties and cloud servers to preserve leakage of side channel information. Due to auditing and regeneration features proposed system is more efficient than existing systems. For data regeneration proxy is introduced it saves the backup copy of uploaded data.

7. REFERENCES

[1] Jingwei Li, Jin Li, Dongqing Xie, and Zhang Cai, "Secure Auditing and Deduplicating Data in Cloud", IEEE transaction, vol.65, no.8, auguest 2016
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted storages," in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 598–609.
[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," ACM Trans. Inform. Syst. Secur., vol. 14, no. 1, pp. 1–34, 2011.
[4] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. 4th Int. Conf. Secur. Privacy Commun. Netow., 2008, pp. 1–10.
[5] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. 14th Int. Conf. Theory Appl. Cryptol. Inform. Secur.: Adv. Cryptol., 2008, pp. 90–107.
[6] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. Comput. Secur., 2009, pp. 355–370.
[7] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in Proc. 7th ACM Symp. Inform., Comput. Commun. Secur., 2012, pp. 79–80.
[8] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in Proc. 28th Annu. Comput. Secur. Appl. Conf., 2012, pp. 229–238.
[9] J. Li, X. Tan, X. Chen, and D. Wong, "An efficient proof of retrievability with public auditing in cloud computing," in Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst., 2013, pp. 93–98.
[10] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in Proc. 18th ACM Conf. Comput. Commun. Secur., 2011, pp. 491–500.
[11] R. Di Pietro and A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication," in Proc. 7th ACM Symp. Inform., Comput. Commun. Secur., 2012, pp. 81–82.
[12] W. K. Ng, Y. Wen, and H. Zhu, "Private data deduplication protocols in cloud storage," in Proc. 27th Annu. ACM Symp. Appl. Comput., 2012, pp. 441–446.
[13] J. Douceur, A. Adya, W. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in Proc. 22nd Int. Conf. Distrib. Comput. Syst., 2002, pp. 617–624.