SECURE DATA AUDITING AND DEDUPLICATING DATA WITH MULTIUSER CLOUD ENVIRONMENT

S.Ghogare[1], Prof.S.K.Sonkar[2]
[1] *M.E., Computer Engineering Department, Maharashtra, India*
[2]*Assistant Professor, Computer Engineering Department, Maharashtra, India*

ABSTRACT

Nowadays, users are more attracted towards outsourcing data to cloud server due to cheap storage cost, efficient management, maintenance, data security etc. Preferably, user outsources their data into encrypted format for privacy concern. There may have chances of de-duplication if similar file is available in cloud storage. Data duplication is one of the important issue which affects on cloud storage. We proposed two secure approaches namely, SecCloud and SecCloud+ for secured auditing and de-duplication in cloud server. On loud server, there may have chances of duplicate data which affects on the management of data. The proposed system check for duplicate files on cloud server before uploading file on cloud by using file tag generation and file block uploading approaches. For verification of data `auditor' is introduced which verifies the data integrity. There are several features introduced in proposed system such as, data encryption, data duplication check and periodic verification of data. For better efficiency map-reduce functionality is used. As a part of contribution, system is design and developed with multiuser environment. User can share files with the other users which is uploaded by them to cloud server. With an experimental set up proposed system proves an efficiency of system in terms of Data uploading, tag generation and deduplication check time of execution.

Keyword: *Auditing, de-duplication, Multiuser System, map-reduce*

1. INTRODUCTION

Users take advantage of the services of cloud server for their heavy data management. Cloud deals with the policy of ``pay-as-you-use” but it may causes to loss of storage efficiency if same content is repeatedly added to cloud server. Also user has to pay for more same content. In the recent survey of EMC it is found that 75 to 78 percent of data on cloud is having duplicate copies. Hence de-duplication check is necessary. But it is not so secure other way round. In case of user uploads data and he is about to know that same data is shared already then it will leak major private and sensitive data. Hence there must be robust system that smartly manages data de-duplication [2]. There must be a system can easily manage data de-duplication. There are several existing systems are available which manages the data but that are not more secured. To manage sensitive data leakages PoW is the proof of ownership protocol is used [3]. In proposed system there are three entities such as, data owner who wish to outsource their data to cloud server, second entity is cloud server which stores and manages the user uploaded data and the third is data auditor which verifies the integrity of user uploaded data. The proposed system works in distributed manners. At client side file uploading protocol is carried out which checks whether if particular file is already stored at cloud or not. PoW protocol is used between client and cloud entity. Data auditor does not have storage facility in it but it verifies status of data integrity on periodic basis if client assigns the permission for it. If data owner uploads the data on cloud via auditor then auditor provided receipt to the client. When user uploaded file come to auditor, it generates the tags of file and then it is submitted to the cloud server. There are three phases included in the task of file upload such as, PoW protocol is run between cloud and client. In second phase, client uploads file through auditor and in third phase auditor generate file tags and upload file along with tags to the cloud. After file uploading, data integrity auditing protocol is also run when user wants to check the integrity status of own data on cloud. For integrity verification, integrity protocol is used which verifies data proofs provided by the cloud. For proof verification, auditor sends the challenge message to cloud server. Cloud response to auditor with proof which is generated by using file content and file tags and try to prove that data integrity. Proof is true, if client / auditors end based on same file tags and file content. There are two objectives of proposed system such as,

[1] Data auditing for integrity purpose
[2] De-duplication checking with secured way

As part of contribution, system is design as multiuser environment, in which multiple users can share and download files from cloud server. In this part owner share particular file with user with access rights. Based on the access rights user can read, write or update particular shared document. User can upload his own file and share with other user. In this case he treats as owner by the system. This approach is most cost effective because if we consider the data uploading protocol then based on file tags proof is generated and based on file tags it is verified hence minimum bandwidth for data transaction is used. Also data integrity checking is required minimum bandwidth as proofs and verification messages are summarized one and transaction with minimum bandwidth is possible. Along with this as part of contribution we use multi-user system. This is also cost effective module as existing protocols and phases are used in it. Hence all features are extended and new functionality of sharing data is added. This sharing is also having some basic data entries and do not consume so much bandwidth. Hence, it is cost effective and extended one.

2. RELATED WORK

Review of literature focused on some previously exist secure data auditing and de-duplication processes / techniques by considering their advantages, disadvantages and features. Following is the summary of the literature study and basic analysis is briefed.

First we focused on the techniques or papers having details about out sourced data integrity auditing mechanism.

J. Li, X. Chen, M. Li, et al [2], proposed Dekey. It is an efficient and reliable convergent key management scheme for secure deduplication. It applies deduplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. They implemented Dekey using the Ramp secret sharing scheme and demonstrate that it incurs small encoding/decoding overhead compared to the network transmission overhead in the regular upload/download operations.

S. Halevi, et al [3], put forward the notion of proof-of-ownership, by which a client can prove to a server that it has a copy of a file without actually sending it. This allows to counter attacks on file-deduplication systems where the attacker obtains a “short summary” of the file and uses it to fool the server into thinking that the attacker owns the entire file. We gave three definitions for security in this setting and three matching protocols, the last of which is very practical.

R. Burns, Ateniese, et al. [4] discussed PDP solution as far as data integrity checking by auditor to the cloud in concern. Uploaded file having n-blocks are verified. Packet Data Protocol is the protocol that verifies that the cloud storage return a file consisting ‘n’-blocks. . In [5], author Burnus and Ateniese uses Packet Data Protocol model for distant data checking. End user who outsourced the data to the cloud can audit or verify the owned data present on semi-trusted cloud without downloading it. It simply generates the proofs about the ownership by randomly sampling the set of file blocks. Hence it is cost effective as file ownership verification is done with minimum bandwidth. In [6] it was proposed to designed simple and secure Packet Data Protocol. This protocol is deal with cryptography having symmetric key. This protocol is not suited for third party verification. Mentioned protocol supports the dynamic outsourcing of data in cost effective manner. It can suitable for more real world approach.

Same kind of work is extended in [7], it is contributed by Dr. K. Manivannan and Dr. T. Nalini, The main motto is to stop data revealing by un-trusted service providers when the data owner distributes their data entries with an error recovery. The distributed scheme supports dynamic data integrity, block updating when block delete and append is implemented. Author aim to reduce the cost of dynamic updates with trusted data transaction innovations. C. Chris, Charalampos Papamanthou et al [8], extend previous work towards DPDP i.e., dynamic provable data possession. They utilized a new version of dictionary based on rank information. In their proposed work they also use rank based RSA-tree to implement DPDP-II with higher probability. They utilized a new version of dictionary based on rank information. In their proposed work they also use rank based RSA-tree to implement DPDP-II with higher probability. In [9], O. Rahamathunisa Begam, T. Manjula et al. introduced PDP for ensuring data integrity in outsourcing of data. The construction of PDP scheme is more efficient as it is implemented in distributed format. It is based on homomorphic verifiable response as well hash index for cooperative PDP scheme to support for dynamic scalability on multiple server storage.

Proof of Retrievability is also the part of data integrity hence literature survey is carried for it also. POR i.e. proof of retrievability system [10], prove to the verifier that client actually storing all clients' data. It gives the complete proof of security arbitrary adversaries in the strongest model. A cryptographic system permits users of outsourced storage service to audit that their data is still available and ready for retrieval if needed. Qian Wang, Cong Wang, Jin Li, et al [11], proposed data security model to enable third party auditor to evaluate the quality of service from independent and objective point of view. Author explores the problem of to provide the public verifiability and dynamic data integrity checking. A POR scheme in [12], required constant number of communication bits per verification and 1/s storage overhead. An authenticated file system is designed to outsource as enterprise class file system to the cloud side. M. Azraoui, K. Elkhiyaoui et al. discussed about POR which integrates the use of randomly produced watchdog with the lightweight privacy preserving word search strategy to gain high retrievability assurance [13].

Following literature survey is carried out for de-duplication also.

A] PoW Proofs of Ownership

Proof of ownership is concept of forward notion by which a user can prove to server that it has copy of the file without sending actual file. It is counter attack on de-duplication of file system in which attacker obtains short overview of the file and utilized it as, to make fool the server that thinking the attacker owns the complete file. CDN attack described in this paper is not much strong as it is convincible hope for [14]. There are several PoW approaches based on the Merkel-hash tree concept. It enables the secure de-duplication at client side and addresses the high entropy case. The Markel hash tree and PoW composed solution can be implementable in small sections. But the Markel hash tree is twice expensive than the SHA256 on the same file. An improved s-PoW is an extreme simplicity model which required a very small room optimization. Hash is required for by server due to independent identification of a file from already stored challenge seed and also to compute the response from client. In this setting, a cryptographic property of hash does not strictly require.

B] Techniques for confidentiality of de-duplicated data

Data de-duplication techniques for private data storage is introduced by W.Keong Ng, Y. Wen et al. in [15]. De-duplication of private data allows client to prove to the server side that who hold the string of the data that he or she is the owner of that data without revealing further information at server side. Secure de-duplication protocol based on the framework assuming that hidden function is collision resilient. A distribute file system known as, Farsite is introduced in [16]. It provides the security and reliability of data by storing encrypted replicas of individual files on different desktop. The replicas are stored in free space may caused file duplication. Cryptosystem enable to identify identical file to integrate even that encrypted with different keys. MLE [17] is Message Locked Encryption technique. In this encryption and decryption key is derives itself from the message. Another is IBE scheme is discussed in [18], to define ciphertext chosen security for an identity based system.

Customer always encrypt their data before upload it on cloud due to the reason of ranging personal privacy. To addressed this problem a key server in introduced in[1] called as, SecCloud with SecCloud+ schema. Both schema supports for integrity auditing and secure de-duplication. It introduced an auditing entity with the maintainance of MapReduced cloud. In secCloud, client's generate data tags before uploading data on cloud for provide better data integrity auditing whereas, SecCloud+ is motivated from the procedure of encryption of data before uploading it on cloud and enable integrity auditing and secured data de-duplication on encrypted data.

3. PROBLEM DEFINITION

To design and develop "a secured data de-duplication & integrity auditing system"

4. PROPOSED SYSTEM

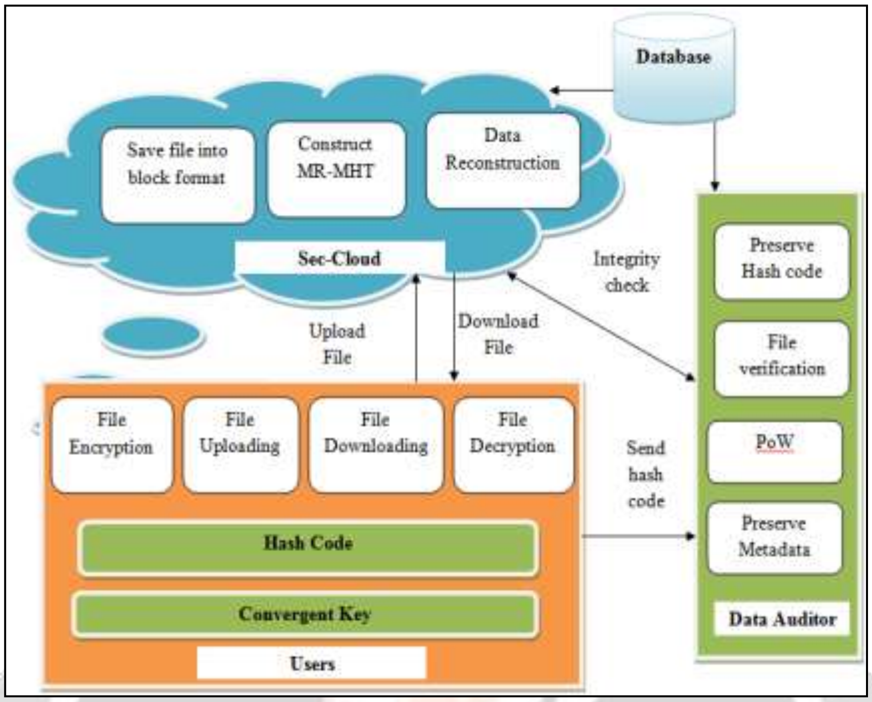

Fig -1: System Architecture

Fig -1 represents three modules of proposed system such as, user, data, auditor and SecCloud. Respective working of each module is given as below:

1. User:
1. Select file
2. Encrypt file: File is encrypted using convergent key.
3. Upload encrypted file on secCloud.
4. Send request secCloud for file download.
5. Download or retrieve file from secCloud using hash code.
6. With the help of keys decrypt file and save it.

2. Data Auditor:
1. Save hash code values.
2. Verify user uploaded files using PoW protocol.
3. Preserve metadata for integrity checking.

3. SecCloud Server:
1. Save user upload file into block format.
2. To enable secure client-side deduplication construct Merkle hash tree.
3. Recover data.

A. Methodology for File Upload:
1. Register user data on SecCloud
2. User Login on SecCloud
3. SecCloud return the identification token T to the user
4. User selects the file to upload and add the group members with whom the file will get shared.
5. File tag generation at user end using SHA-1
6. Send tags to Auditor using HTTP connection
7. Auditor checks privileges of user

If access Privilege Check passes then system will allow deduplication check else gives error message

In deduplication check, it matches the file tag with existing file tags

Case I: If tag matches run proof of ownership (PoW) and share link with other users

Case II: If no file tag matches then it will check deduplication at block level Generate file blocks

1. Generate block hash code
2. Send block hash code to Auditor using HTTP connection

Case I: If Partial Duplication found:

1. It runs proof of ownership for partial no of blocks for new blocks generate convergent key and return token +key + block matching file information to the user.
2. User encrypts unmatched block data using convergent key Upload token+encrypted data and file info to SECCLOUD.
3. Auditor saves the token Run proof of ownership Share link with other users

Case II: If no duplication found

1. Generate convergent key and return token +key User encrypt file block data using convergent key & AES algorithm.
2. Upload token +encrypted data to SEC-CLOUD Auditor saves the token.
3. Run proof of ownership.
4. Share link with other users.
5. Cloud generates MR-MHT for file and uploads metadata to auditor.
6. Auditor saves metadata

B. **Methodology for File Download**:

1. Register user data on Auditor
2. User Login on Auditor
3. Auditor return the identification token T to the user
4. User Ask for file to download to Auditor
5. Auditor checks the privileges of user.
6. If user has privileges it returns file info + decryption key to the user
7. User sends file info and token to SECCLOUD
8. SECCLOUD verifies the token and return file blocks to the user
9. User decrypts the block using AES decryption algorithm and generates the original file

C. **Data Verification:**

1. Verification request and send auditor
2. User generates generate challenge message & send to cloud
3. Cloud read the block data & generate MR-MHT using MR-MHT tree
4. Generate metadata for challenge message & send to Auditor
5. Auditor verifies the proof by matching generated proof with saved metadata
6. Notify user for verification result

5. ALGORITHMS

5.1 AES Algorithm:

Input:

Plain text message m in Byte [] , Key k

Output:

Cipher text message in byte []

Processing:

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[]

5.2. AES Decryption:

Input: Cipher text message C in byte [], Key k

Output: Plain text message m in Byte []

Processing:

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 ⊕
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 ⊕
8. Copy State[] to output[]

5.3 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

6. MATHEMATICAL MODEL

System S can be defined as:

S = {U, S, A, K}

Where,

1. U= {UI, UF ,UO } is user

UI = {UI1, UI2, UI3, UI4, UI5}, A set of user input From User	UI1 = Registration Details of User
	UI2 = Login Details of User
	UI3 = File
	UI4 = Folder to save file
	UI5= Request for Auditing
UF = {UF1, UF2, UF3, UF4, UF5, UF6, UF7, UF8, UF9, UF10, UF11, UF12, UF13,UF14}, A set of	UF1 = Registration
	UF2 = Login
	UF3 = Generate File Blocks

Function	UF4 = Generate Hash Value
	UF5 = Upload Hash Value
	CCF6 = Generate Convergent Keys
	UF7 = Upload Key To Key Server
	UF8 = Encrypt File Blocks
	UF9 = Upload File Blocks To Cloud Server
	UF10 = Call Auditor For Data Auditing
	UF11 = Download File Blocks
	UF12 = Download Keys
	UF13 = Decrypt File Blocks
	UF14 = Save File
UO ={UO1, UO2, UO3}, A set Of Output	UO1 = Success Note
	UO2 = Downloaded file
	UO3 = Audit Report

2. CS = {SI, SF, SO} is A Seccloud Server

SI = {SI1, SI2, SI3, SI4}, A set of Input	SI1 = Encrypted File
	SI2 = Download Request
	SI3 = Challenge Message
	SI4 = Regenerating Codes
SF = { SF1, SF2, SF3, SF4, SF5, SF6 }, A set Of Function	SF1 = Save Blocks
	SF2 = Generate Merkel Hash Tree
	SF3 = Generate Metadata
	SF4 = Generate Proof
	SF5 = Download File
	SF6 = Generate Data from Codes
SO ={ SO1, SO2,SO3,SO4 }, A set Of Output	SO1 = Success/ Failure Note
	SO2 = Metadata

	SO3 = Blocks For Downloading
	SO4 = 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
AO ={AO1,AO2 }, A set Of Output	AO1= Challenge Message
	AO2 =Verification Result

4.KS = {KI, KF, KO } is A Key Server

KSI = {KSI1, KSI2, KSI3}, A set of Input	KSI1 = User Details
	KSI2= Key
	KSI3 = Hash code
KSF = {KSF1, KSF2, KSF3, KSF4, KSF5}, A set Of Function	KSF1 = File Access Rights
	KSF2 = Save user details
	KSF3 = Save aggregate Key
	KSF4 = Save Hash value
	KSF5 = Check Deduplication
	KSF6 = Define POW
KSO ={KSO1, KSO2, KSO3}, A set Of Output	KSO1 = Key
	KSO2 = De-duplication result
	KSO3= User File Details

7. EXPERIMENTAL SETUP

There are three entities present in proposed system such as, user, Seccloud and data auditor.

System is implemented on java-jdk 1.7.0 platform. Apache-6 and mysql-5.6 is configured on same system for server side setup. Netbeans 8.0.1 IDE is used for implementation of client side system. The client side GUI is designed using swing components. Eclipse indigo is used for cloud server system implementation.

Dataset:
Synthetic Dataset: For system testing we have generated synthetic dataset. It contains the files of different type of extension such as, xml, json, txt, java, .cs. etc.

8. RESULT TABLES AND DISCUSSION

Table 1: Time estimation values for proposed system

File size(in KB)	**File deduplication check time**	**Block deduplication check time**	**File upload check time**
200	1.397	11.545	23.311
400	2.698	22.519	43.232
600	2.947	49.167	94.176
800	7.402	122.528	245.488
1000	9.106	166.026	322.737

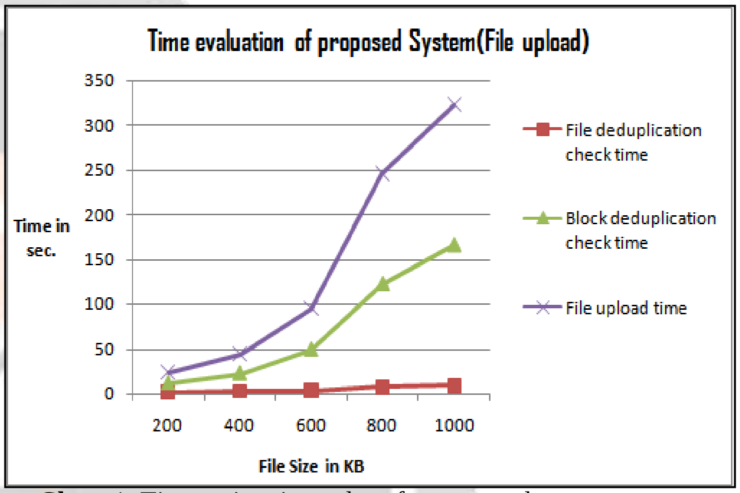

Chart 1: Time estimation values for proposed system

Table 1 represents the time estimation values in terms of file de-duplication, block de-duplication and file upload check time. For testing we have used 200, 400, 600, 800 and 1000 KB of files. According to observation, time required for block de-duplication check and block upload is more than time required for file de-duplication.

Chart 1, represents the time estimation values in graphical format. In this X-axis consists of file size in KB and Y-axis consists of processing time on second.

Table 2: Time estimation values for existing system

File size(in KB)	**File deduplication check time**	**Block deduplication check time**	**File upload check time**
200	2.412	8.249	34.74
400	1.991	36.337	171.715
600	3.234	50.441	271.715
800	12.123	53.838	312.1
1000	7.944	52.43	349.368

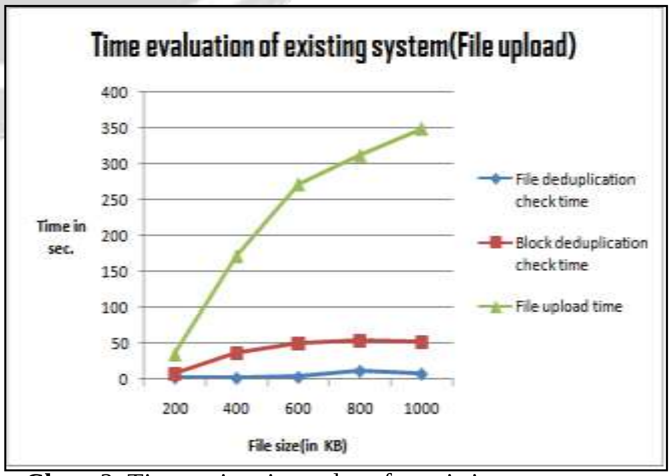

Chart 2: Time estimation values for existing system

Table 2, represents the time estimation values in terms of file de-duplication, block de-duplication and file upload check time. For testing we have used 200, 400, 600, 800 and 1000 KB of files.
Chart 2 represents the time estimation values in graphical format for existing system. In this X-axis consists of file size in KB and Y-axis consists of processing time in second.

Table 3: Partial File de-duplication

File size(in KB)	25%	50%	75%	100%
200	32.52	29.52	24.18	0.43
400	68.2	64.43	50.39	2.98
600	101.72	95.23	76.33	3.45
800	192.44	167.18	143.07	4.76
1000	260.64	231.21	174.02	6.02

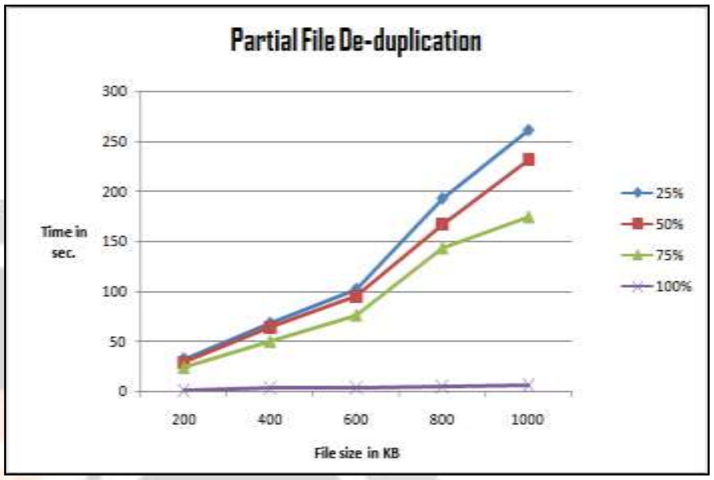

Chart 3: Partial De-duplication check

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

Table 4: Performance evaluation

File size(in KB)	File de-duplication time	Block de-duplication time	Block upload time
200	1.397	11.545	23.311
400	2.698	22.519	43.232
600	2.947	49.167	94.176
800	7.402	122.528	245.488
1000	9.106	166.026	322.737

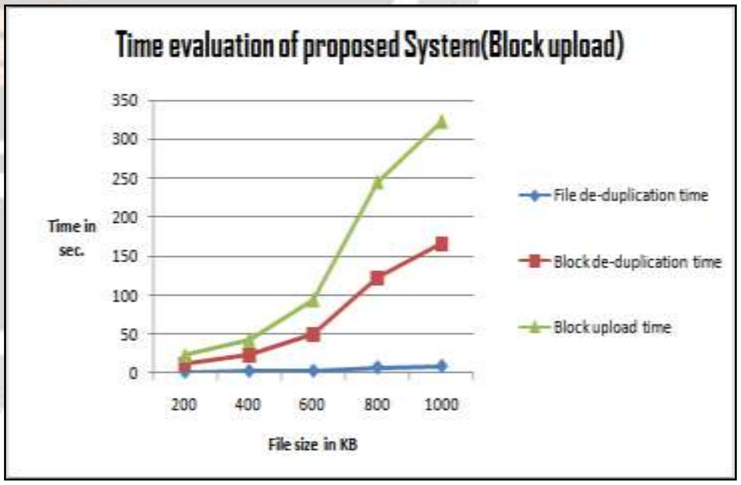

Chart 4: Block uploading

Table 4 represents the system performance in terms of file de-duplication, block de-duplication and block upload time. As per analysis, time required for block de-duplication check and block upload is more than time required for file de-duplication.
Chart 4, represents the performance evaluation in graphical format for proposed system. In this X-axis consists of file size in KB and Y-axis consists of processing time in second.

Table 5: Comparative Analysis

File size(in KB)	Existing system deduplication time	Proposed system deduplication time
200	34.74	23.311
400	171.715	43.232
600	271.715	94.176
800	312.1	245.488
1000	349.368	322.737

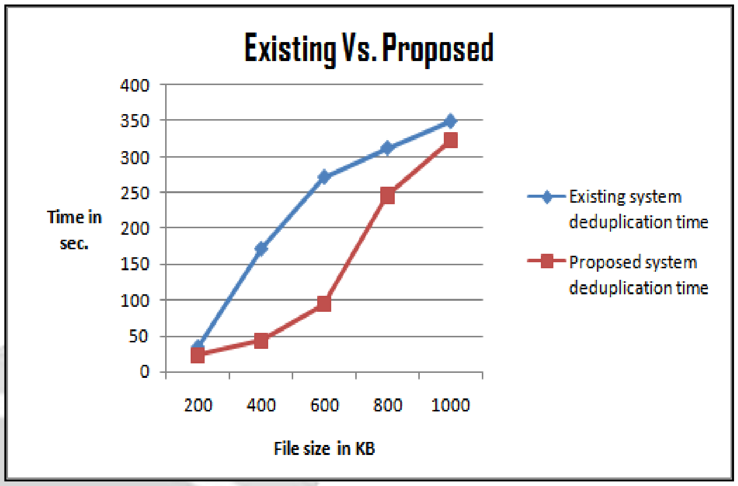

Chart 5: Comparative Analysis

In table 5 comparative analysis between proposed and existing systems is given. Comparison is analyzed in terms of block uploading time. As per observations proposed system required less timing than existing system because in proposed system we utilized block tags as a convergent key for block encryption purpose hence it reduces the processing time as well as saves the space required for convergent key. It predicts that proposed system is space and time efficient as compared to existing system.
Graphical format of system comparison is given in **Chart** 5 in which X-axis consists of file size if KB and Y-axis consists of time in seconds.
Table 6: Comparative Study

Paper	**File level deduplication**	**Block level deduplication**	**Convergent Key**	**MHT-Tree**	**Sharing**	**Space efficient convergent key**	**Data auditing**
Block Level De-duplication Check	Y	Y	Y	N	Y	N	N
Secure Distributed Deduplication	Y	Y	N	N	N	N	N
Secure auditing and De-duplication	Y	Y	Y	Y	Y	N	Y
Proposed Solution	Y	Y	Y	Y	Y	Y	Y

Table VI represents the comparative analysis between existing and proposed systems. Proposed system is more efficient than the existing due to features like, space efficient convergent key, data auditing, MHT tree construction etc.

9. CONCLUSIONS

We proposed approach of data de-duplication and auditing. There are several techniques available for secure data deduplication and integrity auditing. But they suffered from certain problems such as, data reliability. There is problem with encryption technique which is used by existing system, which required different cipher-texts for different users to share identical data. Another problem is integrity auditing as cloud usage traditional way to transformed data through internet and stored it in any random domain. Data de-duplication is one of the recent technologies / techniques in cloud storage in current market trends that avoid such data duplication caused by privileged as well as non-privileged user. It enables companies, organizations to save a lot of money on data storage, on bandwidth to transact data when replicating it offsite for disaster recovery. We proposed two types of techniques such as SecCloud and SecCloud+ to preserve data integrity.

10. ACKNOWLEDGEMENT

A very firstly I gladly thanks to my project guide Prof. S. K. Sonkar, for their valuable guidance for implementation of proposed system. I will forever remain a thankful for their excellent as well as polite guidance for preparation of this report. I would sincerely like to thank to HOD Prof. R. L. Paikrao and other staff for their helpful coordination and support in project work. Also I am thankful of Principal of AVCOE, Sangamner**,** Prof. M.A.Venkatesh for their kind support and motivation for proposed research work.

11. REFERENCES

[1] Jingwei Li, Jin Li, Dongqing Xie, and Zhang Cai "Secure Auditing and Deduplicating Data in Cloud" VOL. 65, NO. 8, AUGUST 2016

[2] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 6, pp. 1615–1625, Jun. 2014.

[3] 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

[4] 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.

[5] 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.

[6] Erway, A. Kupc € ¸u, C. Papamanthou, and R. Tamassia, € "Dynamic provable data possession," in Proc. 16th ACM Conf. Comput. Commun. Secur., 2009, pp. 213–222.

[7] H. Wang, "Proxy provable data possession in public clouds," IEEE Trans. Serv. Comput., vol. 6, no. 4, pp. 551–559, Oct.-Dec. 2013

[8] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. Knowl. Data Eng., vol. 20, no. 8, pp. 1034–1038, Aug. 2008

[9] 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.

[10] 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.

[11] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in Proc. 7th ACM Symp. Inform., Comput. Commun. Secur., 2012, pp. 79–80.

[12] 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.

[13] M. Azraoui, K. Elkhiyaoui, R. Molva, and M. Onen, "Stealthguard: € Proofs of retrievability with hidden watchdogs," in Proc. Comput. Secur., 2014, pp. 239–256.

[14] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 6, pp. 1615–1625, Jun. 2014.

[15] 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

[16] 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.
[17] M. Bellare, S. Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,” in Proc. Adv. Cryptol, 2013, pp. 296–312.

[18] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, “Message-locked encryption for lock-dependent messages,” in Proc. Adv. Cryptol., 2013, pp. 374–391.