International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Policy and Attribute Based Deduplication System

K .Keerthana, E.Sowmiya, N.M.Saravanakumar

Abstract*: Cloud storage is a platform where we can store and access the data through internet. Many Social and business applications are backing up their data in cloud. So, the requirement of cloud storage increases day by day. But most of the files present in the cloud storage are redundant. Removing redundant files in cloud is one way to manage the storage space requirement. For that purpose, we apply File-level data deduplication technique. In this paper, a method for deduplication is proposed which is named as policy and attribute based dedupe system. This system includes access control over files to enhance the security. It also reduces the storage space and increase the security of user data.

Keywords* : Deduplication, File-level data deduplication, Ownership verification, Cloud storage.

I. INTRODUCTION

Cloud computing offers a new way of Information technology services by rearranging various resources (e.g., storage, computing) and providing them to users based on their requirements. It has various desirable properties, such as scalability, elasticity, fault-tolerance and pay-per-use. Thus, it has become a promising service platform. The most important and popular cloud service is data storage service. Cloud users upload personal or confidential data to the data center of a Cloud Service Provider (CSP) and allow it to maintain these data. To make data management scalable in cloud computing, deduplication technique has been used. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Deduplication can take place at either the file level or the block level [14]..

II. LITERATURE SURVEY

In [1] the author divides the proposed scheme in three different segments one is File upload phase. In this phase the user will send upload request to CSP.The upload request contains the tagF (tag for the file), user ID and access policy. The CSP uses the tag F and the access structure *T* to check whether a duplicate file is present in the storage. If there is no duplication, then perform the "First Upload" module [1] to upload the file in CSP. If the duplicate file exists, it enters into the next phase named as "Deduplication" module. Consider the file is already uploaded in CSP by User0. And now User1 is trying to upload the same file. Then the CSP first performs PoW protocol with User1.If User1 pass PoW protocol then CSP add the user ID to the file record and send tagF to the User1.If User1 fails in PoW then the CSP will reject the request from User1. Final phase is File download in which the user send file download request to CSP. The download request contains user ID and tag for the file. The CSP will verify whether the user Id is present in the file record .If it is present the CSP will forward the cipher text along with authorized convergent key material (ACKM) to the user. To decrypt the cipher text the user needs convergent key which can be obtained from ACKM. Using convergent key the user can decrypt the cipher text and obtain their requested file.

In [2] the author describes a novel scheme for deduplication integrated with access control. This scheme contains encrypted data upload, token generation, duplication check, ownership verification and deduplication as main aspects to remove duplicate files in cloud storage. Initially, the user will generate data token by using SHA1algorithm and send it to cloud storage and token comparison will take place at storage to find the duplicate files. When a result of comparison is positive, then the user is verified by authentication party which is known as ownership verification. It uses elliptic curve cryptography [17] to validate the user. If it is a real data owner, the key to access the file will be sent to them. When no duplicate files are found [i.e., the result of comparison is negative] then the user can directly upload their file in cloud storage.

In [3] the author proposes a method for deduplication using attribute based encryption [ABE].First, each CSP user will generate key pairs and get certificates for their public keys. When user sends the data packet to cloud storage, the CSP will verify the certificates of user. Only when the verification result is positive the hash value [generated by using SHA 1algorithm] present in the data packet will be compared to discover the duplicate file. When a duplicate hash value is found at that time the request is forwarded to data owner. The data owner will verify the user by using RSA algorithm to find that he really holds the data or not. After ownership verification the key will be sent to user to access the file.

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Mrs.K .Keerthana*, Information Technology, Vivekanandha college of Engineering for Women, namakkal, India. Email: k.keerthanakarthikeyan@gmail.com
Ms.E.Sowmiya, Information Technology, Vivekanandha college of Engineering for Women, namakkal, India. Email: eswaramoorthysowmiya@gmail.com
Dr.N.M.Saravanakumar, Information Technology, Vivekanandha college of Engineering for Women, namakkal, India. Email: saravanakumar2008@gmail.com

When a hash value is unique, the user can save their data packet in cloud storage without any verification..

In [4] a scheme for authorized duplication check in hybrid cloud was proposed. Here the private cloud server maintains a table which contains user information such as user public

key and their privilege set. Primarily, the user should authenticate in private cloud which help the private cloud to identify their corresponding privileges. Then the user will generate a file tag by using SHA 1 algorithm and send to

private cloud for encryption. After that the encrypted file tag will be send to storage for duplicate check. When duplication occurs, the Proof of ownership protocol is executed by user at cloud storage. After successful verification the pointer of that file and Proof from CSP will be sent back to the user. Finally the proof and privilege is send to private cloud & after verifying proof the file token is generated by private cloud using HMAC-SHA 1 algorithm and it is sent back to the user. When no duplicates occur the proof from CSP is returned to the user and it is verified by private cloud as stated above. At last the user will upload the encrypted file to the storage.

In [5] the system proposed to handle deduplication is similar to the method used in [3]. This system uses Proxy re encryption (PRE) for data deduplication In this method the certificates of user is verified first then the signature present in the data packet will be verified to find the duplicate files. If no duplicate is found the data packet will be stored at cloud. If else the request is forwarded to authentication party for ownership verification. It uses RSA algorithm for ownership verification. If the user is verified by RSA then the key to access the file will be send to user and the corresponding deduplication records will be added to cloud storage or else their request will be cancelled.

In [6] the author developed a scheme for deduplication integrated with dynamic ownership management. Here the cloud server will set a binary tree for its users. Each user is represented as leaf in binary tree. When the user wants to upload their data in cloud they will generate & send the encrypted file, encrypted key and the tag value to the storage and the tag is verified to find the duplicate. When duplicates occur, the id of the users who uploaded the same data is added to form a group. After that, obtain a group key from the binary tree which is used to store the key values of users. Now encrypt the cloud data with group key and the encrypted group key is sent back to valid data owners. When no duplicates occur, the cloud server creates a new group and inserts the id of the user to that group. Then the encrypted file and encrypted key uploaded by that user is saved in cloud server.

In [7] First, the hash value of the file is calculated by using SHA 2 algorithm and it is saved as Check Block where as Cipher Block contains the encrypted file which is encrypted using AES key shorter than prime modulo p. Then encrypt AES key by user public key is saved as Enabling Block and it is encrypted by multiplicative asymmetric homomorphic encryption using elgamal algorithm. At last the user computes the second hash value of the file and it is multiplied by the AES key under the modulus p which is known as Converting Block. Then the cipher structure is uploaded by the user which contains four blocks namely enabling block, check block, cipher block and converting block. By verifying the cipher block in CSP [Cloud Service Provider] the duplicates can be identified. If it is duplicate, the enabling block is converted and it is saved in CSP. If not then the entire cipher structure will be saved.

In [8] the author has proposed three different schemes to remove duplicate files in storage. The author categorizes the upload process into two types. They are first upload and subsequent upload. In Scheme I and II the user authenticate with key server and then send the hash value. If hash value is not found, the key server will return null then it is first upload. Now, the user will generate data encryption key b using random number. After that the data encryption key is encrypted by AES encryption algorithm and sent them to Key server. Finally, the user should authenticate with storage and they can store their encrypted data. If hash value is already present in storage then the key server will return the encrypted data encryption key(S) to user then it is subsequent upload. Only valid user can obtain data-encryption key (R) from S. Then they can authenticate to storage to store and access their files. In Scheme III XOR [13] and permutation functions are used to calculate the Key values.

In [9] the author define a method for deduplication which support both file level and block level deduplication. The user sends the tag value for the file to the storage for duplicate check. If duplicate is found, then the user should prove the storage that they really own the data. If ownership verification is passed then the storage returns the pointer of that file to the user. If duplicate is not found, the tag value for each block will be sent for duplicate check. Similarly if a duplicate block is found then the data owner is verified. After that the pointer for that block is sent back to the user. If no duplicates were found, the entire file will be uploaded at CSP.

In [10] the author proposes a method for deduplication which does not need any additional independent servers for key generation. Initially, the user will generate a short hash value by using SHA 256 algorithm and send it to storage to find the duplicates. If duplicates found, use PAKE protocol [Password Authenticated Key Exchange] [12] to verify that the user really hold the same data. If yes, the users will receive an encrypted key to access the storage data. If no, the user upload request will be cancelled. When no duplicates are found, then the user can directly upload their data in cloud storage. In [11] the author describes a scheme for deduplication which is based on data popularity. When the user sends the index value and encrypted file to the Index Repository Service [IRS] to verify that the user data is popular data or unpopular data. If IRS returns the unchanged index, it is popular data after that the deduplication request is sent to storage from IRS. The Storage verifies the index and then adds the user id to the group. When the index doesn't match with already stored index the user request will be cancelled. If IRS return changed index then it is unpopular data after that the user will doubly encrypt the data before saving it in cloud.

III. PROBLEM STATEMENT

In existing systems, the user will send the encrypted form of data to storage system.

This leads to arise difficulties in finding duplicate files. For instance, if two users encrypt the same file with different algorithm then two different encrypted forms of data will be obtained for a same file. When it is uploaded the storage contains two encrypted files for a single data.

The other issue in existing system is even the authorized users can access only the files uploaded or updated by them. The user cannot directly request a file owner to access their files. This is another drawback of existing systems.

IV. PROPOSED SYSTEM

In policy and attribute based dedupe system, the policies are framed in order to eliminate duplicate files at storage. Policies are defined over a set of attributes [15]. Based on the policies the attributes are verified. All the attributes involved in that policy should satisfy it. If not, then the system fails to satisfy that policy. In addition to that access control over user files are also included in this system. It enhances the security of user data present in storage. Our system contains four modules which are shown below at Fig. 1.

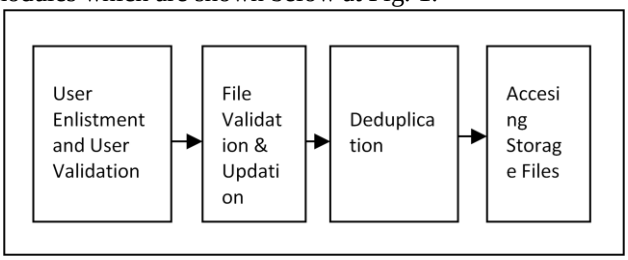


Fig. 1.Architecture diagram for policy and attribute based dedupe system

A. User enlistment and User validation

During enlistment phase, the user will enroll their name and credentials. After enlistment their details will be saved in server as shown in Fig. 2 and they will become an authorized user. Only then they can access our system. During User Validation phase, the credentials entered by the user is verified to ensure that it is a valid user or not. In case of a valid user they can enter into our system for saving and retrieving their file in storage. If else, the user is not allowed to access our system

- **Policies for user enlistment.**
  During enlistment the user should enter a unique id name and password when compared to already enlisted user's id and passwords.

New idname != Existing idname(i) where i=1 to n (1)

New Password!= Exiting password(i) where i=1 to n (2)

- **Policies for user validation**
  Only enlisted users can save or access a file from Storage location

(1) && (2) –> *true* → Storage Access (3)

If it is an invalid user then their login request will not be proceeded

(1) && (2) –> *False* → *Request Cancelled* (4)

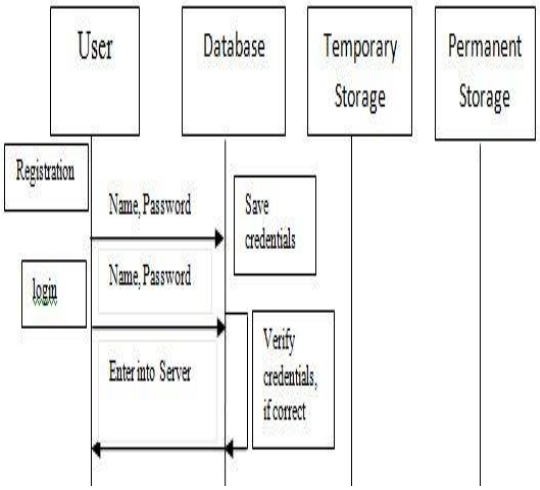


Fig. 2.User enlistment and User validation

B. File validation and updation

During this phase, the user will enter their file name and its description to upload their file into storage. If the file name already exists in storage, the user will get notification from the server. In that case, the user should rename that file before uploading it. Then the server will generate token value for the file by using SHA 256 algorithm. The java code for SHA 256 algorithm that produce token value is given below

```
public String generateToken(File file) throws
FileNotFoundException, IOException,
NoSuchAlgorithmException {
        MessageDigest digest =
MessageDigest.getInstance("SHA-256");
        FileInputStream fis = new FileInputStream(file);
        byte[] byteArray = new byte[1024];
        int bytesCount = 0;
while ((bytesCount = fis.read(byteArray)) != -1) {
          digest.update(byteArray, 0, bytesCount);
        };
        fis.close();
        byte[] bytes = digest.digest();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < bytes.length; i++) {
          sb.append(Integer.toString((bytes[i] & 0xff) +
0x100, 16).substring(1));   }
return sb.toString();      }
```

- **Policies for file validation and updation**

User should enter file name and file description to upload file. Then the file will be saved in temporary storage location and generate token for verifying that it is a unique one.

Enter [name && description] → *Save [file]* → *temporary Storage* → *generate (token [file])* (5)

Only Unique files can be saved in permanent storage location.

newfile name ! = ExistingFile name(i) && newfile token ! = ExistingFile token(i) →*move (newfile)* → *Permanent Storage* (6)

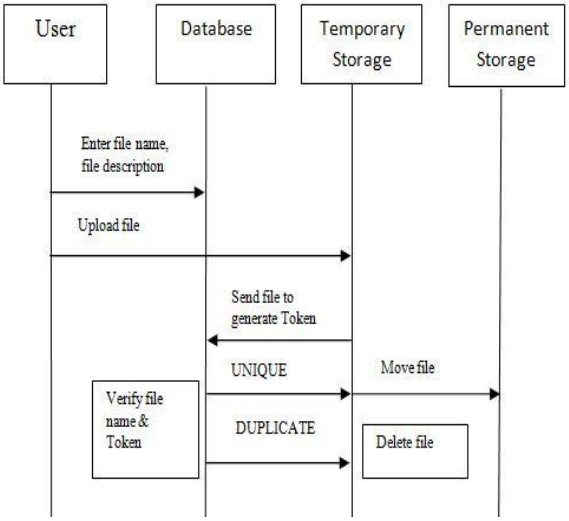


Fig. 3. File validation and updation

C. Deduplication

The Fig. 4 shows a detailed view of eliminating duplicate files. When user uploads the file it will be stored in a temporary location. First the name of the file will be compared with the file name which is already present in storage. If it matches, the user will get the notification to change their file name to upload their file. Based on the file content the token value will be generated for that file with the help of SHA-256 hashing algorithm. Next, the value of the token will be compared with the value stored in the database. If it matches, the file stored in temporary location will be deleted and request the user to search for the file. When the user finds the required file, they can request the file owner to provide access to that file. Therefore, a file with similar name or with similar content cannot be saved in our storage system. Thus our storage system completely eliminates the duplicate files.

- **Policies for deduplication**

When authorized user try to save a redundant file [which has same file name but different file content] then a notification will be sent to user to change their file name.

newfile name == ExistingFile name(i) && newfile token ! = ExistingFile token(i)→delete (newfile)→ notify user → change (newfile name) (7)

When authorized user try to save a redundant file [which has same file content and different file name] then their request will be cancelled.

newfile name != ExistingFile name(i) && newfile token == ExistingFile token(i) →delete (newfile) → notify user (8)

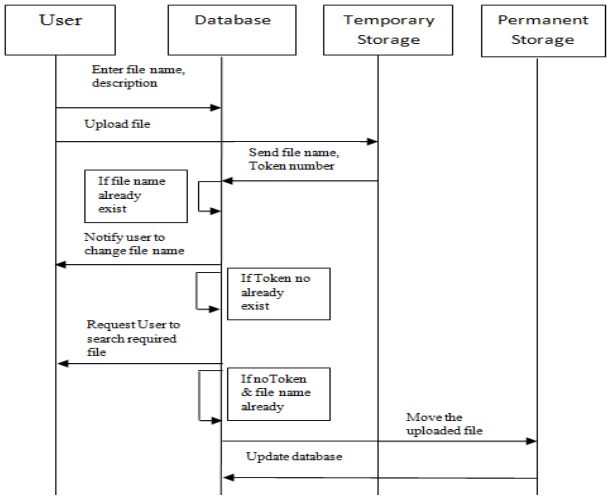


Fig. 4. Eliminating Duplicate files

D. Accessing storage files

During this phase, the user can search the file by using the file name or by file description. The search result will list all the related files. The file description will describe the content of the file and it helps the user to find the correct file. If the user is a file owner of that file, they can access that file directly as shown in Fig. 5. Otherwise the user can request the file owner to provide access to that file. If the file owner accepts the user request then the user can access the requested file. If the file owner rejects the user request, the user cannot access or download that particular file. Here access control of the files is managed by the file owners and the main advantage of our system is any authorized user can request any number of files that they are required. So, without uploading / updating the files the user can access and download files when those file owners accept the user request.

- **Policies for accessing storage files**

Only file owner can access a file directly from the storage location.

Idname == filename.fileuploadedby→ access file. (9)

An authorized user can search for files using file name and file description.

(1)&&(2) →*true* →*File Search* →*Enter [name || description]* (10)

An authorized user can request a file owner to provide access to their files.

(1)&&(2) → *true* → *File Search*→*[Idname != filename.fileuploadedby]* →*request[filename.fileuploaded by]* (11)

If a file owner rejects the request from an authorized user then the link to access that file will not be provided to that user and vice versa.

response [filename.fileuploaded by] →*reject* →*access cancelled* (12)

response [filename.fileuploaded by] →*accept* →*File access* (13)

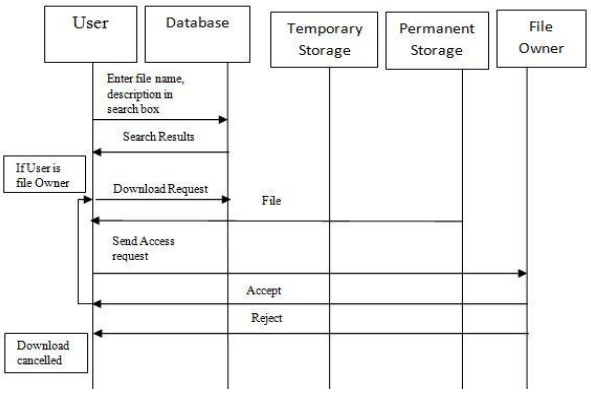


Fig.5 Accessing storage files

V. OUTPUT

This section contains the screenshots of eliminating duplication in policy and attribute based dedupe system. Initially, the user keerthu is uploading her resume in the storage which is shown in Fig. 6. The token value for the file will be generated and it is compared with the hash values present in the database to find the duplicate files. Since the file uploaded by keerthu is a unique file [no matching hash values found], the file gets uploaded into the storage.

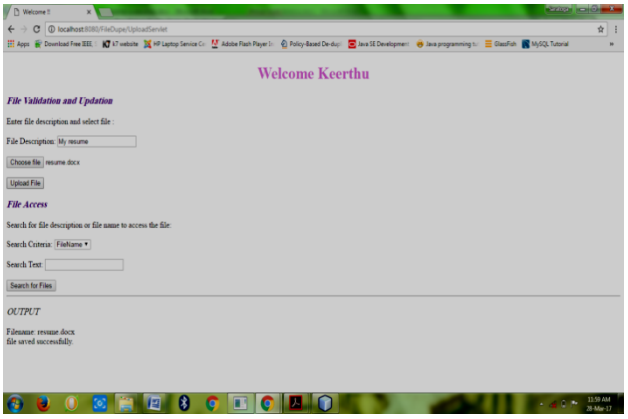


Fig.6 Saving unique file in storage

In Fig. 7, the user rajee tries to save the same file [uploaded by the user Keerthu]. Then the token value for the file is generated and it is compared with the database. Since the file is already uploaded by keerthu. The database will contain the matching hash value. Now the server identifies that the file uploaded by rajee is a duplicate file. Then server will notify the user and cancel the upload request.

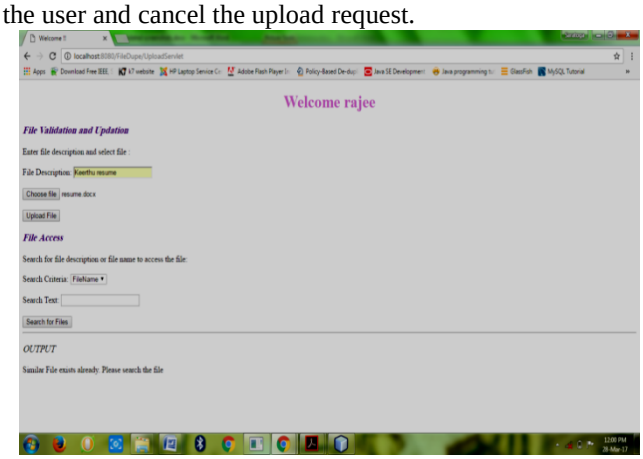


Fig.7. Eliminating duplicate file [same content]

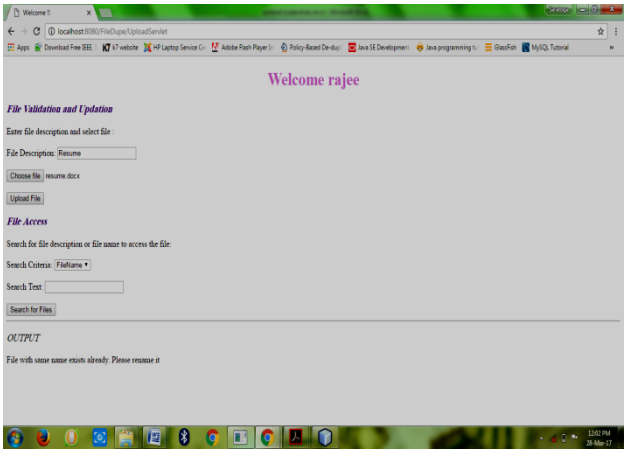
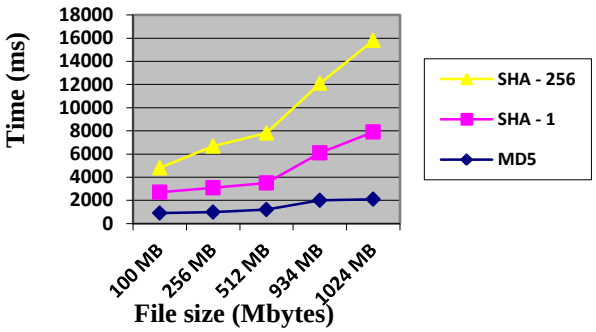


Fig.8. Eliminating duplicate file [same file name]

In Fig. 8, the user rajee is trying to upload the file which has the same file name but different content [as compared to keerthu's file]. Then the server will request the user rajee to rename their file before uploading it. This shows that our system accept only the files which has unique file name and unique content. This leads to eliminate the duplicate files in storage.

VI. PERFORMANCE ANALYSIS

This section illustrates the comparison between efficiency of token generation process and storage space required by existing system and Policy and Attribute based deduped System. In Fig. 9 we have compared three different algorithms for token generation process. They are MD5, SHA-1 and SHA-256. MD5 generate 128 bit length of tokens, SHA-1 algorithm produces a hash of 160 bit length and SHA 256 algorithm generates 256 bit length of tokens. When the length of the token increases it is difficult to attack using brute force attack and the time to generate tokens also increases gradually. MD5 is faster than SHA algorithms [16] but it provides less security than SHA 1 and SHA 2. Among these three algorithms SHA 256 has better security and it take more time to generate tokens. For better protection of user data here we have chosen SHA 256 to generate token.



VII. RESULTAND DISCUSSION

Data Size	Reduced Storage Space		
	Policy &attribute based dedupe	*Cp-ABE based dedupe*	*PRE based dedupe*
62GB	29.2 GB	41.8 GB	46.5 GB

156 GB	123.2 GB	135.8GB	140.5 GB
556 GB	523.2 GB	535.8GB	540.5 GB
998 GB	965.2 GB	977.8 GB	982.5GB

In Storage Space requirement, the space required by Policy & Attribute based deduped System is 17.3% lesser than PRE based deduped System and 12.6% lesser than CP-ABE based deduped System which is shown in the Table 1. It is because we upload the file in temporary storage location and compared the token value with database. And then the file is moved to permanent storage only when it is unique. But in existing Systems the encrypted file is directly uploaded into permanent Storage when its token value is unique. When a file is encrypted by different algorithms then it is possible to obtain two different token values for a single file. As a result, the policy based dedupe system eliminates nearly 32.8% of duplicate data, CP-ABE eliminates 20.2% of duplicate data and PRE eliminates 15.5% of duplicate data from the uploaded data.

VIII. CONCLUSION AND FUTURE ENHANCEMENT

In this paper, we have resolved two main problems in deduplication one is finding duplicates among encrypted data and the other is access the file without uploading it. To resolve this, we have uploaded the data into temporary storage instead of uploading encrypted form of data into temporary or permanent storage location. So finding duplicates becomes easier. And the search option is included in our dedupe system to find the required file present in the Storage. Finally, the performance analysis evidence that the performance of policy and attribute based dedupe system is higher than the existing systems. In future, based on the policies encryption keys will be generated and issued to the users for accessing the storage files.

REFERENCES

1. Taek-Young Youn, Nam-Su Jho, Kyung Hyune Rhee and Sang Uk Shin "Authorized Client-Side Deduplication Using CP-ABE in Cloud Storage", In: *Hindawi Wireless Communications and Mobile Computing*, Vol.2019, Article ID 7840917.
2. Z.Yan, W.Ding, X.Yu, H.Zhu, and Robert H. Deng, "Deduplication on Encrypted Big Data in Cloud", In: *IEEE Transactions On Big Data*, Vol.2, No.2, pp. 138- 150, April-June 2016.
3. Z.Yan, M.Wang, Y.Li, and Athanasios V. Vasilakos, "Encrypted Data Management with Deduplication in Cloud Computing",In: *IEEE Cloud Computing*, pp. 29- 35, March/April 2016.
4. Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, and Wenjing Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication",In: *IEEE Transactions on Parallel and Distributed Systems.*
5. Zheng yan, Wenxiu Ding, and Haiqi Zhu, "A Scheme to Manage Encrypted Data Storage with Deduplication in Cloud",In: *Springer International Publishing Switzerland*, pp. 547–561, 2015.
6. J.Hur, D.Koo, Y.Shin, and K.Kang, "Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage",In: *IEEE Transactions on Knowledge and Data Engineering*, pp. 1041-4347, 2016.
7. Chun-I Fan, Shi-Yuan Huang and Wen-Che Hsu, "Encrypted Data Deduplication in Cloud Storage",In: *10th Asia Joint Conf. on Information Security*, pp. 18- 25, 2015.
8. R.Miguel, Khin Mi Mi Aung, and Mediana, "HEDup: Secure Deduplication with Homomorphic Encryption", pp. 215 – 223, March/April 2015.
9. M.Wen, K.Lu, J.Lei, F.Li, and J.Li, "BDO-SD: An Efficient Scheme for Big Data Outsourcing with Secure Deduplication",In: *The Third* International Workshop on Security and Privacy in Big Data, pp. 214 – 219, 2015.
10. J.Liu, N.Asokan, and B.Pinkas, "Secure Deduplication of Encrypted Data without Additional Independent Servers", pp. 874 – 885.
11. J.Stanek, and L.Kencl, "Enhanced Secure Thresholded Data Deduplication Scheme for Cloud Storage",In: IEEE Transactions On Dependable And Secure Computing, 2016.
12. S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks",In: IEEE Computer Society Symposium on Research in Security and Privacy, pp. 72-84, 1992.
13. Shu Qin, Ren., et al, "Homomorphic Exclusive-Or Operation Enhance Secure Searching on Cloud Storage",In: Proc. of International Conf. Cloud Computing Technology and Science (CloudCom), 2014.
14. K.Keerthana, C.Suresh Gnanadhas and R.T. Dinesh Kumar, "A Survey on Managing Cloud Storage using Secure Deduplication", In: Institute of Integrative Omics and Applied Biotechnology (IIOAB),Vol.7, pp. 656- 666, Dec 2016.
15. http://crypto.stackexchange.com/questions/17893/what-is-attribute-based-encryption.
16. 16.http://msdn.microsoft.com/en-us/library/ms 078 415.aspx
17. 17.https://bithin.wordpress.com/2012/02/22/simple-explanation-for-elliptic-curve-cryptography-ecc/

AUTHORS PROFILE

Mrs.K.Keerthana, received the B.E. degree in Computer Science and Engineering in 2015 from the Anna University, Chennai and the M.E. degree in Computer Science and Engineering in 2019 from the Vivekanandha College of Engineering for Women(Autonomous), Namakkal. She is an Assistant Professor in the Department of Information Technology, Vivekanandha Engineering College for Women (Autonomous), Tiruchengode. She is a ISTE member.

Ms.E.Sowmiya, received the B.E. degree in Computer Science and Engineering in 2015 from the Anna University, Chennai and the M.E. degree in Computer Science and Engineering in 2019 from the Vivekanandha College of Engineering for Women(Autonomous), Namakkal. She is an Assistant Professor in the Department of Information Technology, Vivekanandha Engineering College for Women (Autonomous), Tiruchengode.She has 2 years of Teaching experience and She is a ISTE member.

Dr.N.M.Saravanakumar, received the B.E. degree in Computer Science and Engineering in 1999 from the Madras University, Chennai and the M.E. degree in Computer Science and Engineering in 2005 from the Anna University, Chennai. He received his Ph.D. degree in the area of Information and Communication from Anna University, Chennai in 2013. He has 18 years of Teaching experience. He is a Professor and the Head of the Department of Information Technology, Vivekananda Engineering College for Women (Autonomous), Tiruchengode. He is recognized as a Supervisor for Ph.D Programme (By Research) in Information Technology for Anna University, Chennai. He is in the editorial board member of International/National Journals. He has published 45 papers in international, national journals and conference proceedings and having Google Scholar citations and h-index.He is guiding more than 10 Ph.D research scholars in various recent domains. His areas of research include Network Security, Distributed Computing and Cloud Computing. He has delivered several special lectures in workshops and seminars.