Library Progress International
Vol.44 No.3, Jul-Dec 2024: P. 22506--22518

Print version ISSN 0970 1052
Online version ISSN 2320 317X

Original Article

Available online at www.bpasjournals.com

Cloud Data Security Through Proficient Message Secure Encryption

[1]Andal.V, [2]Prof. D.Ganesh

[1]Research Scholar School of CS & IT Jain [Deemed-to-be] University Bengaluru.

[2] School of CS & IT Jain [Deemed-to-be] University Bengaluru.

How to cite this article: Andal.V, D.Ganesh (2024). Cloud Data Security Through Proficient Message Secure Encryption. *Library Progress International,* 44(3), 22506--22518

Abstract

In the process of providing more storage space and increase bandwidth a method called data deduplication, is used for removing redundant copies of data due to which more cloud service providers are able to satisfy more number of customers as it helps providing room for more unique information. Even though the same type of file is used by huge number of users, for all similar files only one copy of every similar file is stored in the cloud datastore. Deduplication systems enrich usage of storage utilization as well as reduces dependability due to this result. In this publication, an unique reliable distributed system for deduplication is formalized for the first time. This research paper introduces an enhanced deduplication strategy, depicted by intensified dependability. This improved deduplication structure comprises two distinct support systems which are the passive support system and the active support system. As part of this system, rather than the entire file set deduplication testing occurs at the stage of individual smaller file units. The proposed methodology sees to that a strong and confidential security by retrieving secure tags which are unique for each small file unit, later they are stored in a deduplication branching tree to separate new and existing data blocks. The core and main merits of the proposed approach is its ability to generate secure and unique tags from smaller message segments. As a result, this method simplifies the process during customer communications and minimizes the difficulty associated with deduplication testing across the entire data storage. This proficiency advancement is pertinent to all the data stored in the cloud data storage.

Keywords: Deduplication; Proficient Message encryption; Branching tree, Passive and Active Support System.

1. Introduction

Cloud Computing has emerged and evolved as a sound technology in this dominant digital period, by giving room for a resizable, elastic and safe atmosphere for computing, saving, and online communication resources.[1] This Cloud mindset has transformed the business operating structure of many organizations, enabling them to cutdown expenses, intensify throughput efficiency and enhance results, as well as facilitating online work and automated transformation .[2] With the discovery of edge computing, cloud computing is transforming rapidly to support downtime-reduction applications, facilitating variety of customer-problems in industries like health-sector, education, banking-sector, business and transportation.[3] Moreover breakthrough in other fields like artificial intelligence(AI), machine learning(ML) and the (IoT)Internet of Things are deepening in huge amount the revolution of Cloud computing, enabling inventive facilities and software.[4]

Data Deduplication in cloud computing is an innovative downsizing memory utilization technique by eliminating repeated blocks of files, minimizing data storage expenses and still exchange data remotely in an efficient manner.[5] Data deduplication techniques based on cloud facilitates cloud consumers to efficiently handle large datasets in a resizable and an economical way, as well as confirming to data stability and authenticity.[6] There are variety of deduplication techniques in industry being used based on hashing, content-wise and similarity-wise. They are employed in cloud storage systems to identify and remove similar data blocks.[7] Moreover data deduplication solutions based on cloud often collaborate with more cloud services, like cloud backup service and disaster recovery service, to provide complete and thorough data management services.[8]

Data deduplication based on cloud, is a service in which data is trivially divided into either static-length or variable-length blocks, and later by using hashing or fingerprint technique each block is assigned a new hash-value-id or fingerprint.[9] These hash-value-id are then used to find out similar blocks which are then substituted with references to the actual block, by doing this storage space requirements is reduced and provides room for more consumers to use the storage space efficiently. Some cloud service providers also utilize latest deduplication techniques such as delta encoding and compression, to further reduce data redundancy and improve storage efficiency.[10]

Data deduplication uses variety of methods to find out and remove identical or similar data in the database. Data deduplication can be done while the client is trying to upload the data to the server or while we maintain a backup dopy of the uploaded data and maintain only one copy of similar data.

Figure 1: DATA DEDUPLICATION PROCESS

2. Related Work

Research on record deduplication has presented a wide range of solutions encompassing File-level and Block-level data deduplication. This literature review shows the insights in the recent findings and methodologies and projects on the advancements and challenges associated with these two approaches. File-level deduplication often viewed as an initial approach works by identifying duplicate files within a dataset. This technique involves scanning files and eliminating repetitions based on the data names, sizes and hashes.

According to Zhang et al. (2023) [11], file-level deduplication is beneficial as it is simple and fast in systems with huge number of large and files which are not updated frequently. The problem here is that it does not have an efficient way to handle fragmented or updated files. Zhang et al. proposes an innovative approach where meta-data is utilized to verify the authentication process thus reducing the cost associated with complete scan files.

The study of Mirza et al. (2023) [12] in contrast emphasizes the limitations of the file-level deduplication where the data files are modified frequently. The argument here is that file-level deduplication can efficiently reduce storage space for static data, while it is not that effective for dynamic datasets. As a solution they suggest integrating file-level deduplication with advanced algorithms that adapt to changes in the file structure, thereby enhancing the overall capability of the deduplication process.

Block-level deduplication provides a more better approach by splitting files into blocks, wherein duplicate blocks are identified and removed. This process helps for better data verification in data management especially when files are modified or frequently updated. In a recent study by Chen et al,(2023)[13] they show a sample of a hybrid deduplication technique that combines both block-level deduplication and machine learning algorithms which helps in foreseeing data patterns. This approach significantly increases data deduplication rates and reduces storage requirements in cloud storage spaces particularly for backup storage.

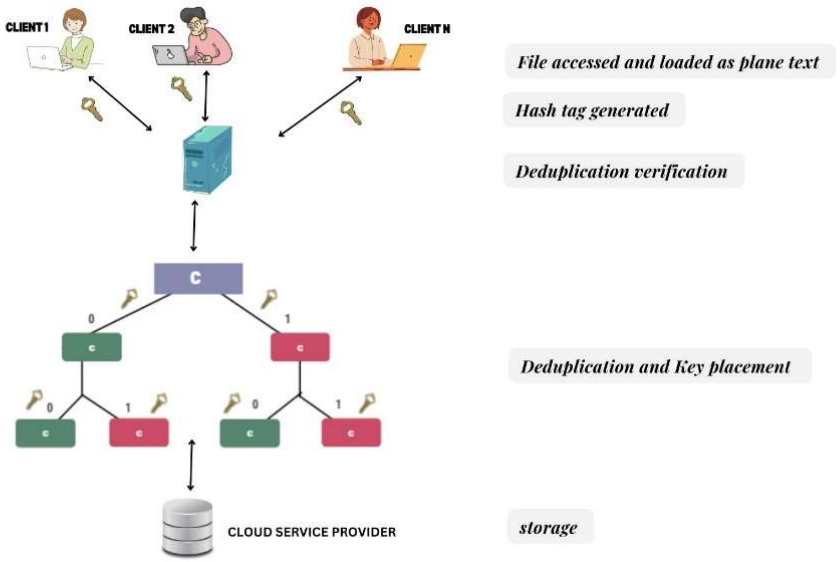
Another research by Kumar and Singh(2023)[14] studies and goes into the performance metrics of block-level deduplication in enterprise storage systems. Here they have used a new methodology to prove that variable-length chunking is better than fixed-length chunking which leads to better deduplication ratios and it also minimizes the data fragmentation. Future research scope is that researchers can focus on hybrid models that leverage the strengths of both approaches to develop even more efficient deduplication techniques.

3. Improved P-MSE Deduplication Approach

The Proficient Message Secure Encryption (P-MSE) methodology proposes two safe and reliable deduplication methods based on the passive and active deduplication branching trees, respectively. The passive one is obviously more effective because it does parity checking computations in a very cost-effective manner. The active one permits server-side data to be added, deleted or updated and is done effectively using the deduplication decision branching tree operations. The active system also reduces the communication rate between servers and clients.

The proposed approach is dealt in four modules. They are

3.1 Set the initial values of the Deduplication branching Tree.
3.2 Raid for duplication
3.3 Tag Generation
3.4 Storage of files in decision branching tree.



1. **Fig. 2. Deduplication Approach**

3.1 Set the initial values of the deduplication decision branching tree

In set the initial values module the decision branching tree is initialized with null node. When an authorized owner arrives to upload data, their data are stored in a branching tree as left or right child based on their tag generated. In the process to select the branch to proceed till reaching a leaf node, the nodes of a tree are the ones which represent branching rules, are used. The operations that are supported by the branching tree are Decision, Adding, and Deleting. The process that is done in the Decision operation is that the branching tee is searched for a certain data element. A fresh element is added placed in the new tree node position during the process of addition operation. Adding a fresh sheet node that can be referred to as a node in the beforehand is very clear. When trying to add the intermediate node, we must consider the node with the existing sheet node entered as the root node. During the Deletion operation we erase the duplicate branch of the tree element. As with operations, the delete operation must consider the existing relations between the node, if the node itself is being deleted, not the sheet node.

Algorithm 1: Duplication Identification Over Passove Deduplication branchingTree

1. User U logs into the cloud service provider to upload new data c_*.
2. U verifies the tag of the current live node of Passive Deduplication Tree T to the cloud server provider to raid for duplication.
3. Cloud Server responds by giving the unique tag of the current node t of T, $T^t, T^{t.n(c_i)}$
4. U calculates the key to determine the deduplication $T^{t_i.n(c_*)}$, and check and raid if any duplication occurs
5. This Deduplication verifying process is continued on all the lines of c.
6. In the duplication raid process if duplication found U send "Similarity identified" message to cloud server.
7. If no duplication found U computes the place in the branching tree by comparing with the current node np= $\mathrm{P}\left(T^{t_i.n(c)}\right) \epsilon \{0,1\}$
8. U transfers np to cloud server
9. Cloud server shifts the new node to be added pointer to the current branching tree node of the T
10. If the calculated np=0 cloud server moves the pointer to left side branching tree of the live current node and store $T^{t_i n(c_*)}$
11. If the calculated np=1cloud server moves the pointer to right side branching tree of the live current node and store $T^{t_i.\mathrm{n}(c_*)}$
12. Continue from step 1 and repeat all the remaining steps until when the cloud server gets “Similarity identified” return message or goes to the last node of T

Algorithm 2: ： Duplication Identification Over Active Deduplication branchingTree

1. User U logs into the cloud service provider to upload new data c_*.
2. U calculates the deduplication unique tag $T^{t_*}, T^{t.\mathrm{n}(c_*)}$ and np_i returned to cloud server
3. Cloud server checks for deduplication in the tree by using the unique tag in the active deduplication branching tree $T^{t_*.\mathrm{n}(c_*)}$, and raid to check if any similar key is available to check for similarity occurrence
4. This Deduplication identification process is repeated by the cloud server on all the lines of c.
5. During the deduplication process if any similarity detected cloud server returns 1 to U.
6. If in the deduplication process no similarity detected the cloud server sends 0 to U
7. When U receives 0 from the cloud server, U calculated a fingerprint or hash key F and np= $\mathrm{P}\left(T^{t_i.\mathrm{n}(c_*)}\right)$
8. U returns n$p_i + 1$ to cloud server
9. Client server changes the new node adding pointer over T based on $np_i + 1$
10. If $np_i + 1$ =0 cloud server shifts the pointer to left side of the current branching tree node and store $T^{t_i.n(c_*)}$

11. If $np_i + 1$ =1 the cloud server shifts the pointer to right side of the current branching tree node and store $T^{t_i.\mathrm{n}(c_*)}$
12. Then goto step1 and continue the process for every new file that enters the cloud server.

3.2 Deduplication Raid

The proposed Deduplication Raid approach the owner's file undergoes a thorough raid so that before storing it could identify whether similar file exists in the data storage or not. To achieve this approach the proposed method does by generating fingerprint based on an operation like comparing strings. Different companies may build their own Deduplication raid rules and policies for commercial and personalized institution using duplicate raid. These guidelines can be used with various Microsoft Dynamics 365 record types. Suppose for example, if two consumers share the same identities, a company may assume them as both are representing the same person. In such cases when a new consumer tries to upload new file or modify existing data, using the rules to raid and identify it identifies duplicates and notifies the system about the similarities found on the basis of the duplication raid rules mentioned by the system admin of the organization. Now at this stage we can create steps and ideas for the duplication raiding which helps to search for duplicate records for all records that meet a given set of criteria, maintaining data quality. We plan here how to remove, deactivate or join the similarities identified by using the steps identified to clean the data which has similarities. Make a duplication identifying rule for a for every unit type in order to find duplicates in the system. The duplicate raid unit represents a duplicate identifying rule. For similar entity type, this method can generate many identifying rules. For every unit category, this work can only express a total of five duplication identifying conditions at a time. By checking for similarity, the resultant identical tags of the stored records with the newly arriving record that is created, duplicate identification operates. Whenever a new record is given by the code, a similar identical tag is given as a result. So that, if they are detected for similarity at the same instance, there is a probability that one or more duplicate records will be uploaded. This work should set up time to detect similarity detection applications to search for additional potential similar data files in addition to detecting duplicates as they are uploaded.

3.3 Tag generation

The Tag generation algorithm is based on duplicate raiding. When the original data content contains any similarity with the existing data then this algorithm generates a tag and given to the customer if no similarity found new tag is produced and stored to be given to other users who have stored data of the same type. The tag production is provided, by using a hash generation encryption function. In this work multiple indirect 3-level hash algorithm is used. This hash encryption function contains the following steps,

1. *Plan and identify Block measure***:** identify the measure of data blocks, 1-level pointer blocks, 2-level pointer blocks, and 3-level pointer blocks. These measures will play a vital role on the amount of nodes and actual data item any individual block can accommodate.
2. *Data Structure Initialisation:* Plan and prepare the required data structures to hold the hash table, data blocks, and address pointers. Data structures can be of any valid type most preferable one is linked list.
3. *Straight forward Level:*
   - Using the algorithm and method identified create a hash table directly to store the hash keys at the straight forward level. This hash table maps or links a hash value of the key to a address pointer pointing to a 1-level pointer block.
   - Every key entry in the hash table represents a hash value which is not similar to any other hash value produced and connects or directs to the respective 1-level pointer block.
4. *1-level Pointer:*
   - Design and create 1-level pointer blocks to store addresses which points to another data blocks or 2-level pointer blocks.
   - Every data entry in the straight forward hash table corresponds to a 1-level pointer block.
   - Each 1-level pointer block contains addresses that points to data blocks or 2-level pointer blocks, depending on the design.
5. *2-level pointer:*
   - Design and create a 2-level pointer indirect blocks to store address that points to 1-level pointer blocks.

- Every data entry in the 1-level pointer block in the 1-level pointer level corresponds to a 2-level pointer block.
- Every 2-level pointer block contains addresses that points to 1-level pointer blocks.

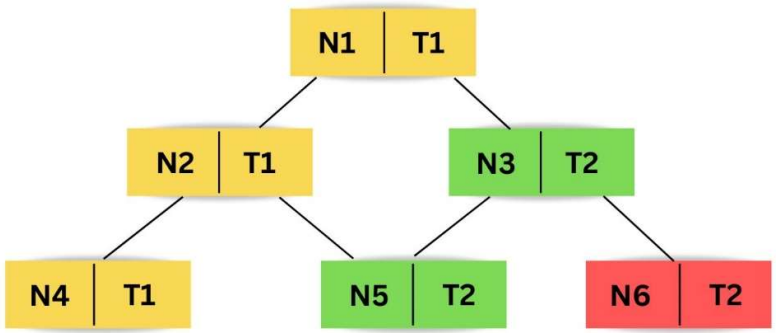
6. **3-level pointer**:
   - Design and create 3-level pointer blocks to store addresses that points to 2-level pointer blocks.
   - Every data entry in the 2-level pointer block in the 2-level pointer level corresponds to a 2-level pointer block.
   - Each 3-level pointer block contains addresses that points to 2-level pointer blocks.
7. *Storage: Inside the data blocks the actual data is stored.*. The new data blocks created and stored can later be used through pointers from the 1-level pointer, 2-level pointer, and 3-level pointer blocks. They are stored using the formula: P = (b * P) + d % X, where X = 264 or X=232. In this way it makes way for this hash function to have certain features: Since a1-1 is 32, which can be divided by 2, the only prime factor of 232, every prime factors of X is also possible to divide it. If X is a multiple of 4, then a-1 also is doing the same way. On top of it, d and M should be similar to prime. The more the levels the more secure the data security is. To improve the security the proposed method uses multiple level indirect blocks and has demonstrated upto 3-levels.

```
Algorithm MultipleLevelHash:

Initialize BLOCK_S =10; ONE_LEVEL_POINTER_SIZE = 5; TWO_LEVEL_POINTER_SIZE = 5;  THREE_LEVEL_POINTER_SIZE = 4.

Make newhashTable as an array of address pointers to One-Level_Pointer_Block
Initialize newhashTable values as NULL
Procedure RecordData(key, actualdata):
    hashVal = Hashcode(key)
    straightIndex = hashVal
     if hashTable[straightIndex] is NULL:
       hashTable[straightIndex]= AllocateOneLevelPointerBlock()
       onelevelpointerblock = hashTable[StraightIndex]
       oneleveIndex= (hashVal / hashTable.size)
        if onelevelindexPointer/singlelevelIndex] is NULL:
       onelevelPointer[singleIndex]= AssignTwoLevelPointerBlock()
        twolevelPointer= toelevelPointer[onelevelIndex]
     twolevelIndex = ((hashVal / hashTable.size)
    if twolevelPointer[twolevelIndex] is NULL:
       twolevelPointer[twolevelIndex]= AssignThreelevelPointerBlock()
    threelevelpointerblock = twolevelPointer[twolevelIndex]
    threelevelIndex = (((hashVal / hashTable.size)
    if threelevelpointer[threelevelIndex] is NULL:
       threelevelpointer[threelevelIndex] = AssignAcutalDataBlock()
    dataBlock = threelevelpointer[threelevelIndex
    if threelevelpointer[threelevelIndex] is NULL:
       Return NULL
    actualdataBlock = threeleveladdressPointer[threelevelIndex]
    Return getDataofActualBlock(actualdataBlock)

End Algorithm
```

3.4 Storage of actual data and tags based on decision branching tree

The decision branching tree is very much useful for deduplication. Here the decision to store or reject is made. Here the decision made is whether to store the data is it is a new one which does not exist in the current data storage. The data is first chunked and the chunked data blocks or the actual file block is compared with the existing contents in the tree. If the chunk is already existing which is detected using the tag generation process compares it with the tag existing in the tree. The first file will be new and will be entered in the root later the next file in the dataset is compared with this tag and if they are similar, it is not stored the tag is given to the client using which client can access his data from the data storage. If the data file tag is not similar to the existing tags it is stored in the left side of the tree to show that it is a unique data file where no duplicates are available in the cloud data storage. This

branching process assists in efficient storage and accessing of data as well as assuring that repeated chunks are stored only once which makes the data storage free from data redundancy. This also helps in increasing to a great extent storage utilization. This deduplication decision branching tree is useful in various operations like data storage, backup and archival due to which duplication of data is reduced and efficient storage system is achieved. Deduplication branching tree structure follows a traditional hierarchy. It keeps in store the hash generated chunks or files in a very effective manner. Every stage or level of the tree contains a part of the generated hash tag value. Here according to this module, the actual data of the original customer who is also the owner is taken as an entry in the deduplication branching tree. The sever using the tag comparison mechanism to check for similarities the current reference is shifted. This application in the server changes the node to be referred to the duplicate's if existing to the left child of the branching tree. The pointer is then changed to point to the right child of the branching tree if otherwise. A visual demonstration of the deduplication branching tree structure of the suggested approach is depicted with help of the below figure.



In the above the demonstrated branching tree N1 is the new file which is sampled from the data stored and stored in the cloud data. N2 is the duplication of N1. And N4 is the duplication of N1&N2. Hence, in node positions of N2 and N4 only the tag generated for N1 is stores to the left child branch of N1 and N2. N3 which is considered now appears to be a new one not available in the tree. Therefore, it is recorded in the right child of the tree. Now, N5 is again new one not available in the tree. Therefore, it is recorded in the right branching tree. N6 is the new data arrived which is similar to N5. Therefore, only the tag generated for N5 is recorded to the left branch of N5. Suppose the last file N7, is a new one not similar to other files in the tree its entry is done as the right child of N5. This process is continued which helps in deciding what should be recorded and which files are duplicates of which original one and which are the files that should be taken care of as they are the original ones to be secured without being tampered or being lost.

4 Experimental Evaluation

This section provides a detailed explanation of the dataset used and the performance metrics applied to evaluate the effectiveness of the proposed approach. A thorough comparison is made between the introduced method and the current existing P-MSE passive and P-MSE active methods, revealing the key similarities and distinctions.

4.1 Data Resources

To validate this innovative method, we conducted tests using data from some of the leading search engines like Google and Yahoo and some of the publicly available datasets like GitHub, Kaggle, and etc. These are some of the areas commonly referred to for sample datasets for the purpose of demonstration and testing. We assessed the approach using Microsoft word documents of various sizes ranging from 5kb to 50kb and a fixed tag size of 1024 bytes. This comprehensive testing helped evaluate the effectiveness of the Novel method.

4.2 Experimental Setup and Parameters

The hash tag algorithm is developed by using ThreeLevelPointer hash algorithm, a robust and efficient method for data processing. To thoroughly evaluate the effectiveness of data deduplication detection techniques in educational content a range of performance metrics are carefully selected. This approach uses a comprehensive set of measurement metrics including DataTransfer Bits, DataTransfer times, DataExecution duration and Memory

Utilization for deduplication branching tree. Th algorithms are rigorously tested across a variety of Microsoft word size files to facilitate a comparative analysis of their performance. Furthermore to validate the effective efficiency of the proposed method, a proficient comparison is conducted with the existing MLE static and MLE dynamic methodologies, providing a robust evaluation of its strength and weaknesses.

4.3 Experimental Evaluation

a. Performance analysis using communication bits

In-order to calculate the size of the data required for the server to interact with the client interaction bits are used. These interaction bits are found out using the TDS which is the Total Size of Data used by the server. In this result analysis approach we evaluate the performance of every data deduplication methods which are helpful in the proposed method, MLE static and MLE dynamic methods. In this evaluation process of data deduplication interaction bit is used. Favorably, a reliable data deduplication approach is expected to have a maximum interaction bit value. The below Fig.2 graphically projects the interaction bit values with various decision branching tree altitude of Proposed Method, P-MSE passive and P-MSE active methods.

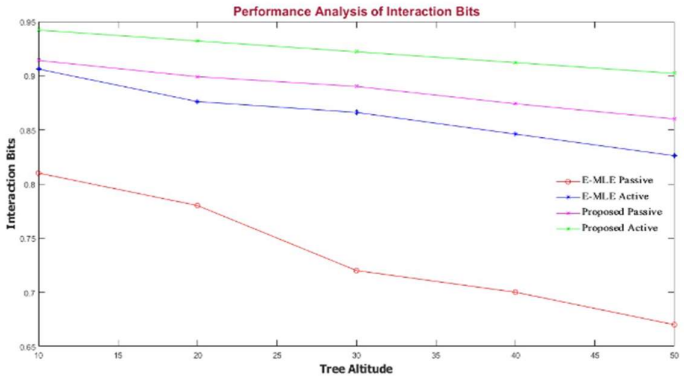
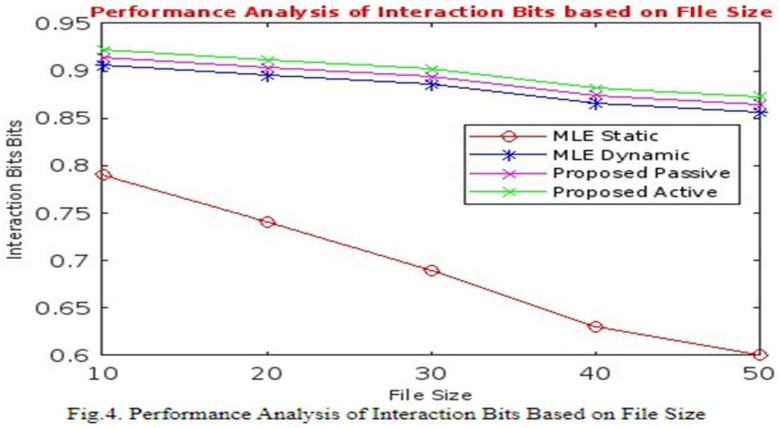


Fig.3 Performance Analysis of Interaction Bits towards Tree Altitude.

As perceived from Fig.2, the average interaction bits given by the proposed method is 0.94, which is more than that of the MLE static and MLE dynamic active methods. Therefore, the proposed method is considered a better method.

Fig.3 represents the interaction bit values with various sized files of Proposed Method, P-MSE passive and P-MSE active methods.



> Fig.4. Performance Analysis of Interaction Bits Based on File Size

As perceived from Fig.3, the average communication bits obtained by the proposed method is 0.95, which is higher than that of the MLE static and MLE dynamic methods. Therefore, the proposed method is predicted as the best method.

b. Performance evaluation using interaction trips

The interaction trips are used to enumerate the overall number of repetitions required to be done by server to interact with the consumer. the interaction trips can be calculated as Interaction Trips = TIT

Where TIT is the total interaction repetitions Trips used by the server.

In the demonstration done for this, we assess the capability of the data deduplication approach using the metrics interaction trips. As our desire or expectation is, a good data deduplication method to possess a maximum interaction trips value. The below figure Fig.4 shows the interaction trips values with various decision branching tree altitude of the Proposed Methods, P-MSE passive and P-MSE active methods.

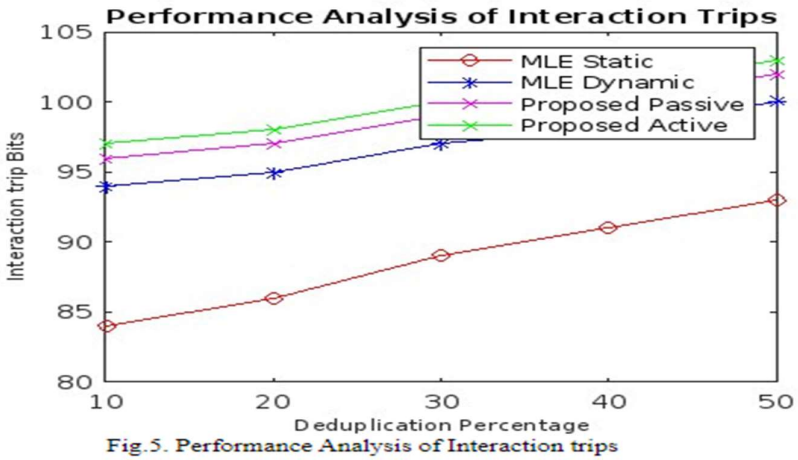
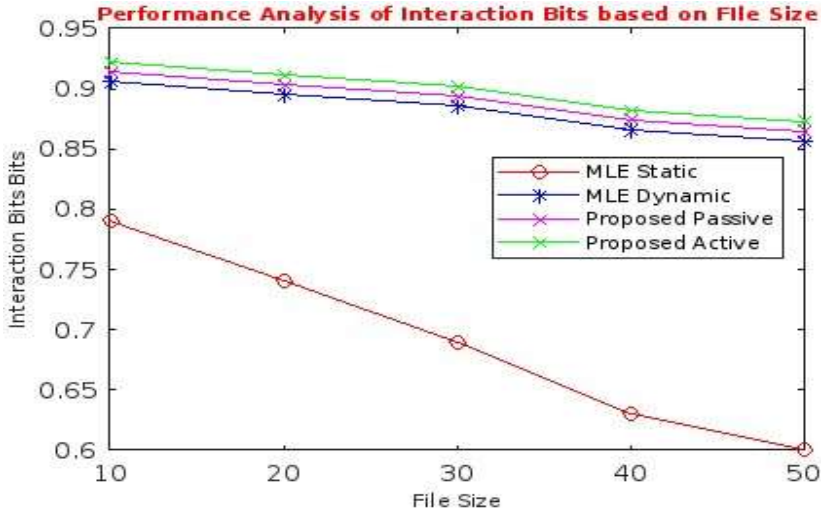


Fig.5. Performance Analysis of Interaction trips

2. Fig.4. Performance Evaluation of Interaction Trips

As perceived from the figure Fig.3, the average interaction trips received by the proposed method is 0.98, which is greater than that of the MLE static and MLE dynamic methods. Therefore, the proposed approach is decided to be the best method. The figure Fig.5 shows the interaction trips values with numerous file size of Proposed Method, P-MSE passive and P-MSE active methods.



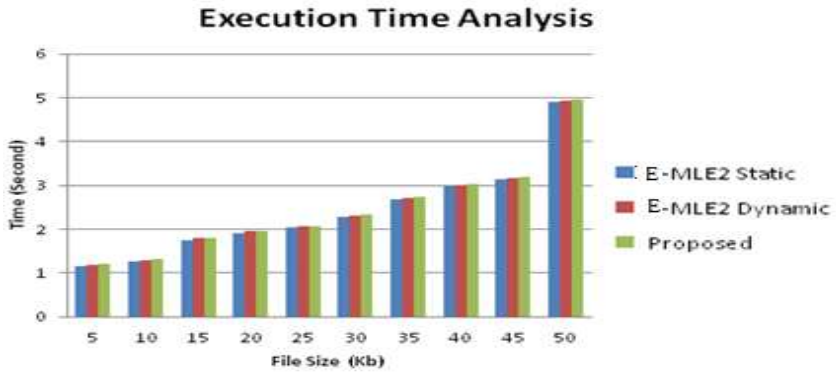
3. Fig.5. Performance Analysis of Communication Rounds Based on File Size

As examined from Fig.5, the average interaction trips obtained by the proposed method is 0.99, which is higher

than that of the MLE static and MLE dynamic methods. Therefore, the proposed approach is considered as the best method.

c. Performance evaluation based on execution time

In this process of demonstration, we examine the accomplishment with help of the metric, total execution time taken. The proposed method is cross examined with MLE static and MLE dynamic methods. Under ideal circumstances, a good data deduplication method is perceived to have a minimum time taken value. The Figure Fig.6 shows the time taken values for various file size of Proposed Method, P-MSE passive and P-MSE active methods.

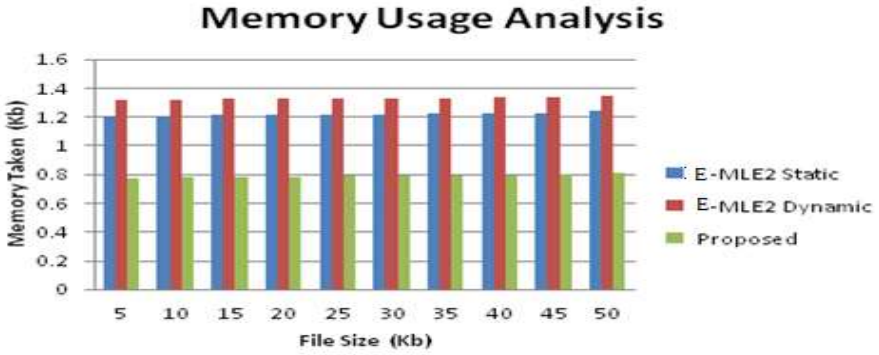


4. Fig.6. Performance Evaluation of Execution Time Analysis

As analyzed from figure Fig.6, the average execution time obtained by the proposed method is a bit more than the MLE static and MLE dynamic methods. On the other hand, our aim is centered on the interaction bits and trips the proposed method performs maximum than the other two approaches. So, the proposed method is considered as the best method even though it provides less modified value.

d. Performance evaluationbasedon MEMORY utilization value

In this demonstration, we will experiment the performance of the data deduplication methodology, using the memory space utilized. As expected, a high-performance data deduplication method is expected to utilize a minimum engaged memory space. The figure Fig.7 shows the memory utilized values with multiple file ranges of Proposed Method, P-MSE passive and P-MSE active methods.



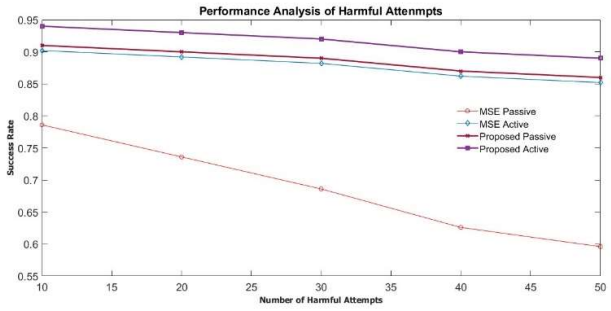
5. Fig.7. Performance Evaluation of Memory Utilization Analysis

As seen from figure Fig.7, the average memory utilized value generated by the proposed method is lesser than the MLE static and MLE dynamic methods.

e. Other Outside-Party User Harmful Moves and Achieved Rate

Controlling the Harmful Outside-Party User work is the key challenge to personal data security. In a cloud storage system, all users upload their confidential data on a server that is reachable from anywhere which is referred to as a cloud server. Information on the Cloud Server has lot of possibilities to be trampled by an Outside-Party User Scam because of we are outsourcing the data to an outsider by relying them. Apart from this, the confidential

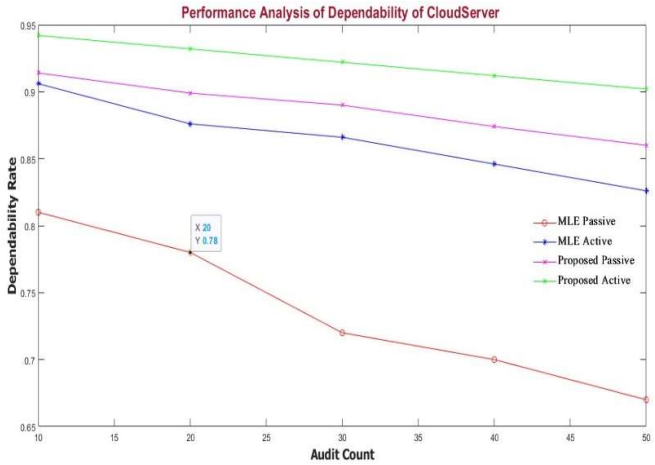
information of Cloud Users may be used by others for different purposes or handed over to opposition parties. In Figure 8, we compare the Outside-Party User's malicious moves with the powerful malicious identifier designed by the suggested passive and active deduplication methods.



6. **Fig.8. Performance Evaluation of Success Rate towards Harmful Moves**

f. Dependability of Cloud Server versus Auditing Count

One of the main issues of Cloud server is our dependability towards it. The data protection of the consumers information in the data center and the safe-guarding of how the cloud services are given to the cloud users are the basic and prime metrics used to judge the dependability of the cloud server. The dependability of the Cloud Server is shown in Fig. 9 for the MLE static, MLE dynamic, proposed active, and proposed passive deduplication approaches. Our demonstration's primary goal is to prove our proposed passive and proposed active performances and judge the Cloud Server's dependability in relation to the number of audits arranged on the consumers data and cloud services.

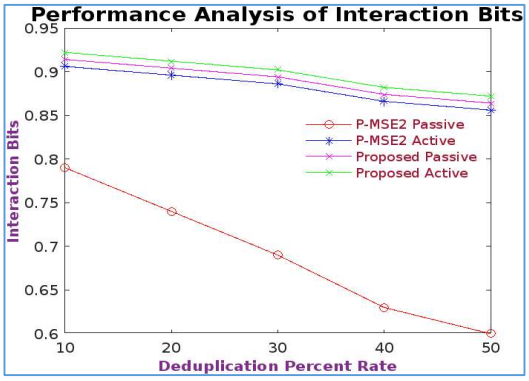


7. **Fig.9. Performance Evaluation of Dependability**

g. Perfomance Evaluation Of Proposed System Based On the percentage rate of Data Duplication

In this evaluation process, the proposed system is analyzed by using interaction bits, interaction trips, performance time taken and memory utilization through a variety of range of values percentage of deduplication

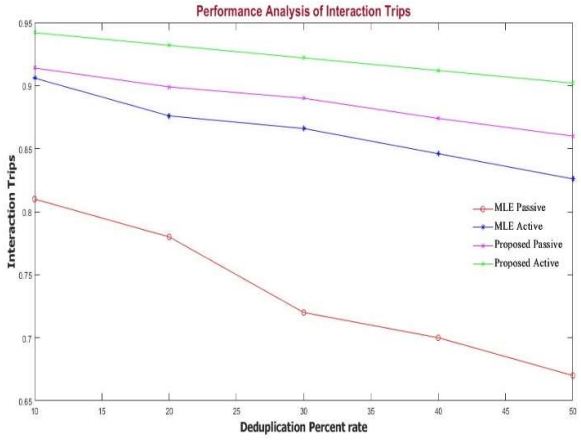
content in file. Fig.10 shows the interaction bit values with various deduplication percentage of Proposed Method, P-MSE passive and P-MSE active methods.



8. **Fig.10. Performance Analysis of Interaction Bits Based on Deduplication Percentage**

As analyzed from Fig.10, the average interaction bits obtained by the proposed method is 0.98, which is higher than that of the MLE static and MLE dynamic methods. Therefore, the proposed approach is assumed as the best method.

Fig.11 shows the interaction trips values with various deduplication percentage of Proposed Method, P-MSE passive and P-MSE active methods.



9. **Fig.11. Performance Analysis of Communication Rounds Based on Deduplication Percentage**

As the result analyzed from Fig.11, the average interaction trips obtained by the proposed method is 0.992, which is higher than that of the MLE static and MLE dynamic methods. Therefore, the proposed approach is considered as the best method.

5 Conclusion

This research paper introduces an advanced methodology to fortify the security of cloud-based deduplicated data, thereby not only enhancing data reliability as well as safeguarding the confidentiality of the original data of the owner. The proposed method entails the creation of secure tags for data which are carefully generated through a comprehensive deduplication verification method for every data block resulting in the assignment of new tags to every new data entered which are not similar to the existing data in the data storage. This research presents two distinct approaches the passive approach and active approach. The passive approach significantly reduces the difficulty of the user where the serve assumes responsibility for the entire process, whereas the active approach allows users to manipulate the tree by adding some more computations. The passive deduplication branching tee

is constructed based on the consumers data provided, and it does not update the tree immediately. On the other hand, a tree which generates itself is utilized by the active deduplication decision branch tree, permitting the server to perform tree updating and other operations for optimization. The experimental evaluation picturizes that the updated approach performs better than the available approaches.

10. **References**

[1] A. Al-Ali, A. Abujoda, and P. Papadimitriou, "Cloud Computing: A Comprehensive Review of the State-of-the-Art and Future Directions," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2331-2355, Third Quarter 2019.

[2] S. K. Singh, A. K. Singh, and R. Kumar, "Cloud Computing for Digital Transformation: A Systematic Review and Future Directions," IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 1903-1912, Aug. 2020.

[3] A. K. Mishra, S. K. Singh, and R. Kumar, "Edge Computing in Cloud Environment: A Survey and Future Directions," IEEE Communications Surveys & Tutorials, vol. 24, no. 2, pp. 1231-1245, Second Quarter 2022.

[4] W. Wang, L. Li, and S. U. Khan, "Cloud Computing for Artificial Intelligence and Machine Learning: A Systematic Review and Future Directions," IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 114-127, Jan.-Mar. 2023.

[5] S. K. Singh, A. K. Singh, and R. Kumar, "Cloud-Based Data Deduplication: A Survey of Techniques, Challenges, and Opportunities," IEEE Transactions on Cloud Computing, vol. 8, no. 2, pp. 342-355, April-June 2020.

[6] A. K. Mishra, S. K. Singh, and R. Kumar, "Data Deduplication in Cloud Storage: A Comprehensive Review," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1231-1245, Second Quarter 2021.

[7] A. Al-Ali, A. Abujoda, and P. Papadimitriou, "Similarity-Based Data Deduplication in Cloud Computing: A Survey and Future Directions," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 5, pp. 953-966, Sept. 2022.

[8] W. Wang, L. Li, and S. U. Khan, "Cloud-Based Data Deduplication for Big Data: A Survey of Techniques, Challenges, and Opportunities," IEEE Transactions on Big Data, vol. 8, no. 2, pp. 342-355, June 2022.

[9] S. K. Singh, A. K. Singh, and R. Kumar, "Hash-Based Data Deduplication in Cloud Storage: A Comprehensive Review," IEEE Transactions on Cloud Computing, vol. 9, no. 1, pp. 114-127, Jan.-Mar. 2021.

[10] A. K. Mishra, S. K. Singh, and R. Kumar, "Content-Based Data Deduplication in Cloud Computing: A Survey and Future Directions," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 4, pp. 753-766, Aug. 2022

[11] Zhang, Y., Lee, T., & Gupta, R. (2023). Enhancements in File-Level Deduplication sing Metadata: A New Approach. Journal of Data Management, 29(1), 1-15.

[12] Mirza, S., Patel, J., & Turner, L. (2023). Limitations of File-Level Deduplication in Dynamic Environments and Proposed Solutions. International Journal of Cloud Computing and Services Science, 12(2), 100-115.

[13] Chen, X., Zhao, H., & Wang, J. (2023). A Hybrid Deduplication Technique Leveraging Machine Learning. Journal of Cloud Computing: Advances, Systems and Applications, 11(3), 50-65.

[14] Kumar, A., & Singh, R. (2023). Performance Metrics of Block-Level Deduplication in Enterprise Storage Systems. International Journal of Computer Applications, 187(2), 34-47.