An Efficient Message Lock Encryption Based Data Deduplication For Efficient Cloud Data Storage

Andal.V

Research Scholar, School of Computer Science & IT, Jain University, Jaya Nagar 9th Block,
Bengaluru, Karnataka 560069, India.
andal.v.srinivasan@gmail.com

Dr. D. Ganesh

Professor, School of Computer Science & IT, Jain University, Jaya Nagar 9th Block,
Bengaluru, Karnataka 560069, India.
ganeshdmca@gmail.com

Abstract

In order to save storage space and upload bandwidth, data deduplication, a method for removing duplicate copies of data, has been widely employed in cloud storage. Even when a file is held by a sizable number of users, there is only one copy for every file saved in the cloud. Deduplication systems enhance storage utilization while decreasing dependability as a result. The idea of a distributed reliable deduplication system is formalized for the first time in this publication. This research paper introduces an enhanced E-MLE deduplication strategy, characterized by heightened reliability. This upgraded deduplication framework comprises two distinct schemes: the static scheme and the dynamic scheme. Within this approach, deduplication verification occurs at the level of individual partial data units, rather than the entire dataset. This methodology ensures robust security by generating unique tags for each partial data unit, which are then stored in a deduplication decision tree. The primary advantage of this approach is its ability to generate tags from individual message segments. This, in turn, simplifies the process during client interactions and reduces the complexity associated with parity testing across the entire database. This efficiency improvement is applicable to the entirety of the data stored in the database.

Keywords:

Deduplication, convergent encryption, message-locked encryption, interactive protocol.

1. INTRODUCTION

Cloud computing is a prominent technology that has helped many organizations save time and money by adding user convenience. So, cloud storage is enormous because organizations can keep their data in practice without worrying about the whole mechanism. Cloud Computing provides key tips to end users, such as saving, sending, and accessing data, regardless of location and operation.

Deduplication techniques are commonly employed for storing information and minimize network and immediate storage by identifying and removing unneeded data. This is due to the progress of digital data. Copying data removes extra data by preserving merely one external design and conveying additional redundant data to that copy as opposed to saving numerous copies of the same information. Since deduplication can help to maximize utilization and conserve space, especially for large redundancy detection systems like preserve file systems, it has drawn a lot of interest from educational institutions as well as from the industry. On the basis of deduplication methodologies, such as server installation, sophisticated document specifications, or block level plugins, numerous redundancy detection systems have been developed. Especially with cloud storage, data acquisition technology is more attractive and necessary to manage the amount of cloud data stored on the cloud, enabling businesses and organizations to input data to external cloud providers. [Arasu *et al.* (2010)]. By 2020, the amount of data generated worldwide is predicted to surpass 40,000,000,000 gigabytes [Arasu *et al.* (2009)]. Today, installation is used by cloud-based storage providers like Dropbox, Google Drive, and Mozy to keep up with the speed and storage requirements of digital clients.

Blocking, comparing, and classifying are the three distinct stages of deduplication. By grouping typical qualities, the blocking phase seeks to minimize the number of comparisons [Christen (2012)]. For instance, in Simple Blocking Methods, any data that share the initial character of a name or feature are entered in the identical block, preventing the formation of pairs of pairs. The comparative stage compares the degree of similarities between pairs that are subject to the same layout using some similar functions (ex. Jaccard, Levenshtein, Jaro [Elmagarmrid et al. (2007)]). Lastly, the categorization step determines whether the pair matches or does not match. This stage can be achieved by selecting the same pair using global threshold, which is generally determined manually [Bayardo et al. (2007)] [Chauduri et al. (2006)] [Wang et al. (2011)] [Xiao et al. (2011)] or obtained using the training-based ranking model.

Although deduplication approaches can save space for cloud providers, it can reduce system reliability. Data reliability is actually the most important issue in the installed database because only one file of each file is hosted by all hosts. If the shared file / folder is missing, the non-proportional data size will be unavailable due to a lack of files that share the file / block. The client information is lost when the database fragment deteriorates with the number of proportional fragments, if the worth of a piece is determined by the quantity of data that will be lost in the event of a block loss. It is crucial to figure out how to guarantee outstanding information dependability in the adapter system. When building the server, the majority of the previous installation systems are only taken into account. However, many polarised platforms and cloud-based storage platforms are built for high dependability by clients and programmes, particularly in the archive system, where data are crucial and need to be stored for a long time. Deduplication storage devices must, therefore, be more trustworthy than high-availability systems. The unprecedented data that is provided to cloud customers also presents a concern for data protection. Generally, the encryption mechanism is used to protect privacy before the data is transferred to the cloud. Because data conversion makes it impossible to replicate, the majority of commercial service providers are unwilling to employ it. The rationale is that conventional encryption techniques, such as symmetric key cryptography and cryptography with publicly available keys, require various clients to secure their information using separate keys. Because of this, copies of the same data made by other users will display different encrypted content. The concept of creating reverse code [Bellare *et al. (2012)*] has been widely proposed and approved to put data secrecy in practice while copying data in order to address the issue of transparency and computerized installation. However, these systems get the secrecy of external data by the value of fault tolerance. Therefore, how to prevent confidentiality and confidence in achieving cloud storage is still a challenge. The following is a brief overview of the related work.

2. RELATED WORK

Research on record deduplication has presented a wide range of solutions encompassing supervised, semi-supervised, and unsupervised approaches. Both supervised and unsupervised strategies rely on expert users for configuring deduplication processes. Earlier researchers required substantial training to establish the primary model within the dataset, as seen in [De Carvalho *et al.* (2012)] and [Wang *et al.* (2011)]. Subsequently, a manual and cost-effective approach was employed for deduplication configuration, as demonstrated in [Bayardo *et al.* (2007)], [Chaudhuri *et al.* (2006)], [Vernica *et al.* (2010)], and [Xiao *et al.* (2011)]. Conversely, the semi-supervised or active approach, closely aligned with T3S, strives to minimize user involvement in configuring the procedure. The aim of the active learning method is to judiciously select an indeterminate subset of the database, thereby enriching the information pool for classification analysis [Elmagarmid *et al.* (2007)]. Historical research has predominantly focused on active learning within binary algorithms to enhance accuracy. In essence, the evaluation of rating quality was conducted through the accurate classification of pairs [Beygelzimer *et al.* (2009] [Cohn *et al.* (1994)]. However, this approach cannot be seamlessly applied to deduplication tasks due to the intricate nature of high imbalance levels, such as pairing instances significantly outnumbering the pairing frequency. As a result, evaluating deduplication tasks requires metrics that assess the precision and recall of recovered exact match segments [Arasu *et al.* (2010)], [Bellare *et al.* (2012)], [Sarawagi and Bhamidipaty (2002)]. To illustrate, [Cohn *et al.* (1994)] introduced a comprehensive learning technique that prioritizes pair selection when the classifier's prediction confidence is restricted. Similarly, in a related context, the researcher behind [Freund *et al.* (1997)] utilized the uncertainty observed among classifiers to pinpoint pairs suitable for labeling. In a different perspective, [Beygelzimer *et al.*(2009)] proposed an active learning methodology known as IWAL. This approach involves marking instances based on the disparity between the current hypothesis, which predicts a pair as matching, and the alternate hypothesis, which predicts the pair as non-matching. The hypothesis is then integrated with prior pairs for further analysis. The communication strategies applied for the independent functioning of ALIAS and Atlas are elaborated upon in [Sarawagi and Bhamidipaty (2002)] and [Tejada *et al. (2002)*]. In a broader context, the Confidentiality Determination Committee, predominantly appointed by users with minimal annotations, often yields results that lack general acceptance. While Atlas employs the Tree Stabilization Tree approach, ALIAS utilizes Naive Bayes and/or SVM classifiers generated at random. In our work, we have integrated ALIAS as a fundamental component. An alternative active learning approach for deduplication is introduced in [Arasu *et al.* (2010)], with a primary goal of augmenting the precision rate. This method establishes an N-dimensional feature space that encompasses analogous functionalities, managed through a combined manual and active partner selection

process employing binary search exploration within the space. Nevertheless, the application of N-dimensional binary search could potentially escalate the magnitude of searches, necessitating a greater manual endeavor [Bellare *et al.* (2012)]. To address this challenge, a strategy known as ALD is introduced in [Bellare *et al.* (2012)], presenting an active and tailored approach for mitigation measures that align with the apparent constraint. Such an approach aims to assess each classifier's efficacy through two-dimensional point estimation. ALD undertakes a binary search within this range to identify the optimal disseminator that adheres to the actual threshold. Dimensions in this context are representative of the rating impact assessed through the oracle. The pairs used for training were curated using IWAL's active learning approach [Beygelzimer *et al.* (2009)]. However, it's important to note that this comparison might not be entirely equitable, given the active learning method's manual constraint restrictions. Nevertheless, we have included ALD as a foundational aspect of our work. In a different endeavor, Corleone [Gokhale *et al.* (2014)] strives to minimize expert interventions within the given scenario. This is achieved through the utilization of a random forest committee to actively extract information with associated tags. The divergence among trees is introduced via random parameter generation, akin to the approach in ALIAS [Sarawagi and Bhamidpaty (2002)]. In contrast to Corleone, the emphasis here lies in the selection strategy for training data that pertains to the scenario involving labeled pairs. This scenario arises when multiple users are involved in labeling pairs within a crowd context. In response, three theories were introduced to establish the cessation point of the learning process. A distinct approach is offered by FS-Dedup [Bianco *et al.* (2013)], which presents various techniques to streamline the deduplication process with minimal user interventions. This is achieved through sorting based on the similarity values. The sorting sequence adheres to a predefined threshold level, encompassing the range of similarity values Subsequently, a tactic is put forth to exclusively label randomly chosen substrings comprised of the most ambiguous pairs at each tier. Moreover, an approach is introduced to configure blockage and ascertain the assortment level. However, it should be noted that FS-Dedup leans on samples that might encompass excessive information, leading to an unwarranted loss of manual labor. In contrast, the T3S strategy we introduced maintains alignment with SSAR training, ensuring each sub-category receives dedicated training. The T3S approach optimally selects a minimized dataset while curtailing the last training module's scope to match that of FS-Dedup, as will be elaborated upon. Shifting focus to the development of training materials, a two-step methodology is proposed by [Christan (2008)]. During the initial stage, the first training set was autonomously formed through the selection of both highly significant and less significant pairs. Subsequently, in the second phase, the initial training set was employed to train a supervised classifier, facilitating the classification and tagging of unmarked pairs, which were then incorporated into the training process. We have also integrated this approach as a fundamental component of our work. Similarly, in [Bilenko and Mooney (2003)], a method is introduced to create a balanced training set, which identifies pairs that demand manual labeling alongside those requiring less manual intervention. In instances where such pairs are lacking, the training set is supplemented through random pair selection. In a parallel vein, a communal deduplication forum introduced a strategy reminiscent of the approach suggested in [Bilenko and Mooney (2003)] for the selection of a training set based on tiered similarity. More specifically, the method involves the selection of pairs with comparable or greater similarity from a predefined tier, which are then labeled by the user. However, these tiers are significantly contingent on the dataset's intrinsic composition, making it challenging for even experts to accurately ascertain the optimal values.

3. IMPROVED E-MLE DEDUPLICATION APPROACH

The two secure deduplication methods that are developed using this suggested methodology are based on the static and dynamic deduplication decision trees, respectively. The static one is significantly more effective because it doesn't require doing pricey pairing computations. The dynamic one enables server-side data addition and deletion and is effective in deduplication decision tree operations. Additionally, the dynamic one will cut down on the number of times clients and servers communicate.

The proposed approach is dealt in four modules. They are

(1) Initialize Deduplication Decision Tree.
(2) Deduplication Checking.
(3) Deduplication based Tag Generation.
(4) Data Storage in Decision Tree.

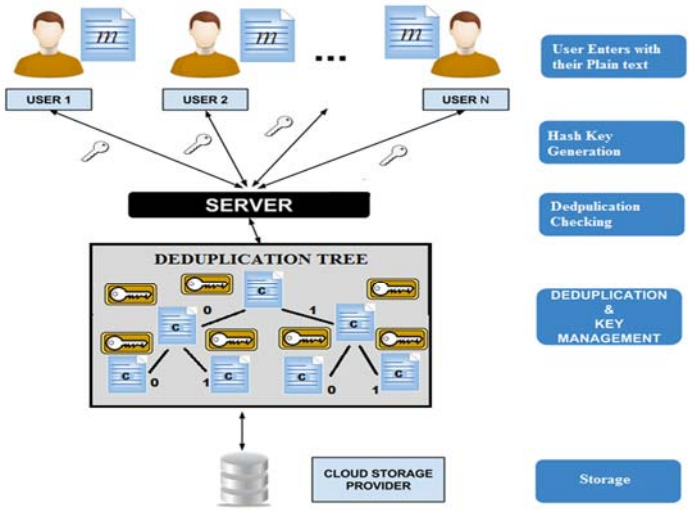

Fig. 1. Deduplication Approach

3.1 InitIalize deduplication decision tree

In Initialization module the decision tree is initialized with empty node. After the data owner arrives, their data are stored in this decision tree as left or right child based on their tag. In order to choose which branch to pursue till reaching a leaf node, the tree's nodes, which represent partitioning rules, are used. Query, insertion, and deletion are the three fundamental operations that decision trees support. In a query operation, the tree is searched for a certain element. A new element is placed in the tree position during an insert process. Inserting a sheet node that can be referred to as a node in the forecast is obvious. When entering the average node, we must take into account associating the node with the node entered as the root node. When deleting something, Remove a specific branch of the tree item. As with operations, the delete operation must consider the connection between the node, if the node is being deleted, not the sheet node.

Algorithm 1: Duplication Checking Over Static Deduplication Tree

(1) The client C login into the cloud for storing new data s_*.
(2) C asks the tag of the current node of Static Deduplication Tree T to the server for checking duplication.
(3) Server send the tag of the current node d of T, $T^{d}, T^{d_0.h(s_i)}$
(4) C computes the deduplication $T^{d_i.h(s_*)}$, and verify if any duplication occurs
(5) Deduplication process is done on each and every line of s_*.
(6) If duplication found C send "duplication found" to server.
(7) Otherwise C computes the position of the current node p= $P\left(T^{d_i.h(s_*)}\right) \epsilon \{0,1\}$
(8) C send p to server
(9) server moves new node inserting pointer to the current node of the T
(10) If p=0 server moves the pointer to left side of the current node and save $T^{d_i.h(s_*)}$
(11) Otherwise server moves the pointer to right side of the current node and save $T^{d_i.h(s_*)}$
(12) Then return step 1. These steps are followed until when the server gets “duplication found” message or reaches the end node of T

Algorithm 2: Duplication Checking Over Dynamic Deduplication Tree

(1) The client C login into the cloud for storing new data s_*.
(2) C computes the deduplication $T^{d_*}, T^{d_*.h(s_*)}$ and p_i send to server
(3) server checks deduplication $T^{d_*.h(s_*)}$, and verify if any duplication occurs
(4) Deduplication process is done by server on each and every line of s_*.
(5) If duplication found server send 1 to C.
(6) otherwise server send 0 to C
(7) When C receives 0 from server, C computes hash key H and p= $P\left(T^{d_i.h(s_*)}\right)$
(8) C send $p_i + 1$ to server
(9) server moves new node inserting pointer over T based on $p_i + 1$
(10) If $p_i + 1$ =0 server moves the pointer to left side of the current node and save $T^{d_i.h(s_*)}$
(11) Otherwise server moves the pointer to right side of the current node and save $T^{d_i.h(s_*)}$
(12) Then return step 1.

3.2 Deduplication checking

In the Deduplication Checking approach the content of owner is checked whether it is duplicated or not. For this purpose, string comparison operation is used. Organizations can develop duplicate detection rules and duplicate detection policies for commercial and bespoke entities using duplicate detection. These guidelines can be used with various Microsoft Dynamics 365 record types. For instance, if a lead and a contact share the same name and phone number, an organization may describe them as being the same person. When a user attempts to add new records or alter existing data, the system warns them about possible duplicates based on the duplication detection criteria the administrator has specified. This activity can plan a duplication detection job to look for duplicate records for all records that meet a given set of criteria, maintaining data quality. By eliminating, deactivating, or combining the duplicates identified by a duplication detection operation, this work can clean the data. Make a duplication detection rule for a certain entity type in order to find duplicates in the system. The duplicate rule entity represents a duplicate detection rule. For the same entity type, this operation can generate different detection rules. For each entity type, this work can only publish a total of five duplication detection rules at once. By comparing generated match codes of existing records with each new record that is created, duplicate detection operates. Each time a new record is produced, a matching code is generated. Therefore, if they are processed at the same time, there is a chance that one or more duplicate records will be produced. This work should schedule duplicate detection jobs to look for additional potential duplicate records in addition to identifying duplicates as they are created.

3.3 Deduplication based tag generation

In the Deduplication based Tag generation, the owner data content contains any duplication then the same tag is provided to the content otherwise new tag is generated. The tag generation is done by using hash function. In this work triple indirect level hash algorithm is used. This hash function have the following steps, a straightforward class of functions that produces a string of seemingly random numbers.

(1) *Determine block sizes***:** Decide on the sizes of data blocks, single indirect blocks, double indirect blocks, and triple indirect blocks. These sizes will impact the number of pointers and data items each block can hold.

(2) *Initialize structures:* Create the necessary data structures to hold the hash table, blocks, and pointers. These may include arrays, linked lists, or other data structures.

(3) Direct level:

- Create a hash table at the direct level. This table maps a hash of the key to a pointer pointing to a single indirect block.
- Each entry in the hash table represents a unique hash value and points to the corresponding single indirect block.

(4) Single indirect level:

- Create single indirect blocks to store pointers to data blocks or double indirect blocks.
- Each entry in the direct hash table corresponds to a single indirect block.
- Each single indirect block contains pointers to data blocks or double indirect blocks, depending on the design.

(5) Double indirect level:

- Create double indirect blocks to store pointers to single indirect blocks.
- Each single indirect block in the single indirect level corresponds to a double indirect block.
- Each double indirect block contains pointers to single indirect blocks.

(6) Triple indirect level:

- Create triple indirect blocks to store pointers to double indirect blocks.
- Each double indirect block in the double indirect level corresponds to a triple indirect block.
- Each triple indirect block contains pointers to double indirect blocks.

(7) *Data storage:* Store the actual data in data blocks. These data blocks can be accessed through pointers from the single indirect, double indirect, and triple indirect blocks. It is stored in the format: Y = (b * Y) + d; // "mod K", where K = 232 or 264. In order for this hash function have certain properties: Because a-1 is 32, which is divisible by 2, the only prime factor of 232, all prime factors of K can divide it. If K is a multiple of 4, then a-1 also is. Additionally, c and M should be somewhat prime.

```
Algorithm TripleIndirectHash:
   BLOCK_SIZE = 8;  SINGLE_INDIRECT_SIZE = 4;  DOUBLE_INDIRECT_SIZE = 4;
TRIPLE_INDIRECT_SIZE = 4.
  Create hashTable as an array of pointers to SingleIndirectBlock
     Initialize hashTable entries to NULL
Procedure InsertData(key, data):
       hashValue = Hash(key)
       directIndex = hashValue
        if hashTable[directIndex] is NULL:
          hashTable[directIndex]= AllocateSingleIndirectBlock()
          singleIndirect = hashTable[directIndex]
          singleIndex= (hashValue / hashTable.size)
           if singleIndirect[singleIndex] is NULL:
          singleIndirect[singleIndex]= AllocateDoubleIndirectBlock()
           doubleIndirect= singleIndirect[singleIndex]
       doubleIndex = ((hashValue / hashTable.size)
       if doubleIndirect[doubleIndex] is NULL:
          doubleIndirect[doubleIndex]= AllocateTripleIndirectBlock()
       tripleIndirect = doubleIndirect[doubleIndex]
       tripleIndex = (((hashValue / hashTable.size)
       if tripleIndirect[tripleIndex] is NULL:
          tripleIndirect[tripleIndex] = AllocateDataBlock()
       dataBlock = tripleIndirect[tripleIndex
       if tripleIndirect[tripleIndex] is NULL:
          Return NULL
       dataBlock = tripleIndirect[tripleIndex]
       Return RetrieveDataFromBlock(dataBlock)

End Algorithm
```

3.4 Data storage in decision tree

The deduplication tree's structure allows efficient storage and retrieval of data while ensuring that duplicate chunks are stored only once. This approach greatly optimizes storage utilization and reduces redundancy. Deduplication trees are used in various data storage systems, backup solutions, and archival systems to achieve storage efficiency and reduce data duplication.

The deduplication tree is a hierarchical structure that stores these hashed chunks efficiently. Each level of the tree represents a portion of the hash value. In this module the content of the owner is saved on the decision tree. Depending on whether there are duplicates, the server moves the tree's current reference. The server shifts the reference to the duplicate's left child if one exists. The current pointer is then moved to the right child if not. The sample deduplication tree structure of the proposed approach is shown in below figure.

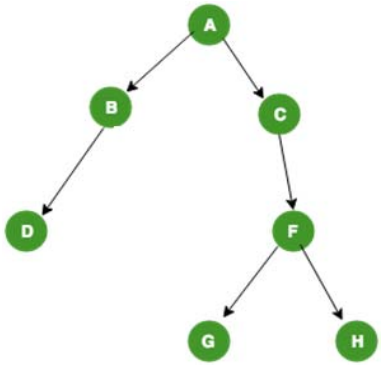

Fig. 2. Deduplication tree structure of the proposed Approach

In the above the tree A is the first file stored in the cloud data. B is the duplication of A. And D is the duplication of A&B. So, in place of B and D only tags are stored as left child. C is original content. So, it is stored as right child. Again, F is also a original content. So, it is stored as right child. G is the duplication of F. So only tag is stored as left child. Finally, H is original content. So, it is stored as right child.

4. EXPERIMENTAL EVALUATION

Within this segment, an intricate elucidation is furnished regarding the dataset employed and the performance metrics leveraged to assess the efficacy of the introduced approach. A comprehensive juxtaposition is drawn between the proposed method and the E-MLE2 Static and E-MLE2 Dynamic methodologies.

4.1 Dataset used

The validation of this novel approach involved conducting tests on data sourced from prominent search engines like Google and Yahoo. This data, commonly utilized in the compilation of students' research endeavors, was employed for assessment. The implementation parameters were duly enumerated, encompassing Microsoft Word documents of assorted dimensions. The file sizes spanned from 5 KB to 50 KB, while the hash tag's size was standardized at 1024 bytes.

4.2 Implementation and parameter settings

The hash algorithm is derived through the utilization of the triple indirect level hash algorithm. To assess the effectiveness of the deduplication detection techniques in educational content, a variety of performance metrics are at hand. This study employs metrics including Communication Bits, Communication Rounds, Execution Time, and Memory Usage for Deduplication Tree to comprehensively evaluate performance. The algorithms were executed across varying sizes of word files for comparative analysis. In order to substantiate the efficiency of the proposed method, a comparison is drawn against the E-MLE2 Static and E-MLE2 Dynamic methodologies.

4.3 Experimental analysis

4.3.1 Performance analysis using communication bits

The communication bits are used to calculate data length needed by server to communicate with the client. The communication bits can be calculated as
Communication Bits = TDL where TDL is the total data length used by the server. In this experiment, we will evaluate the contribution of each data deduplication approaches which are used in the Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods. To evaluate the performance of this data deduplication approach, the communication bit is used. Ideally, a good data deduplication approach is expected to have a high communication bit value. Fig.2 shows the communication bit values with various tree height of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

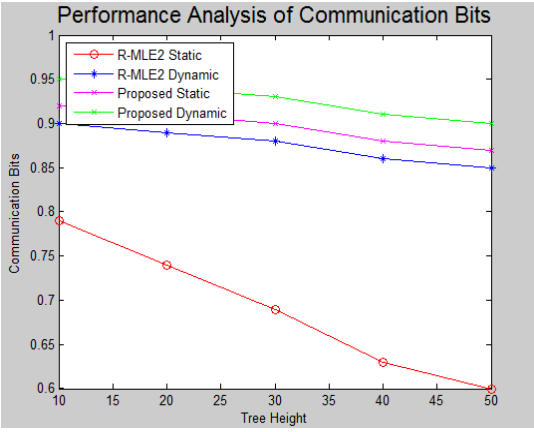

Fig.3. Performance Analysis of Communication Bits

As observed from Fig.3, the average communication bits obtained by the proposed method is 0.93, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So the proposed method is considered as the best method. Fig.4 shows the communication bit values with various file size of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

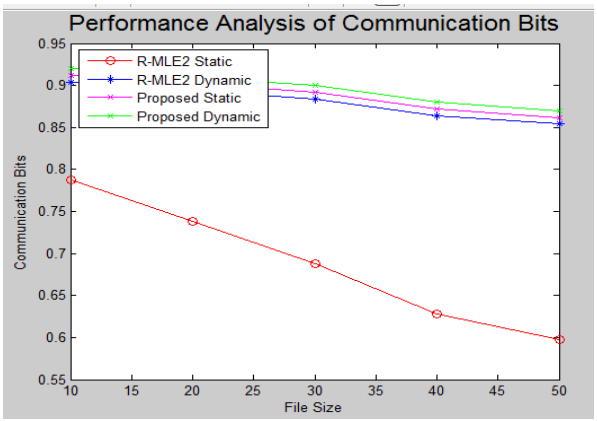

Fig.4. Performance Analysis of Communication Bits Based on File Size

As observed from Fig.4, the average communication bits obtained by the proposed method is 0.95, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So, the proposed method is considered as the best method.

4.3.2. Performance analysis using communication rounds

The communication rounds are used to calculate total no of iteration needed by server to communicate with the client. the communication rounds can be calculated as Communication Rounds = TCI

Where TCI is the total iteration used by the server.

In this experiment, we analyse the performance of the data deduplication approach using the communication round. Ideally, a good data deduplication approach is expected to have a high communication rounds value. Fig.5 shows the communication rounds values with various tree height of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

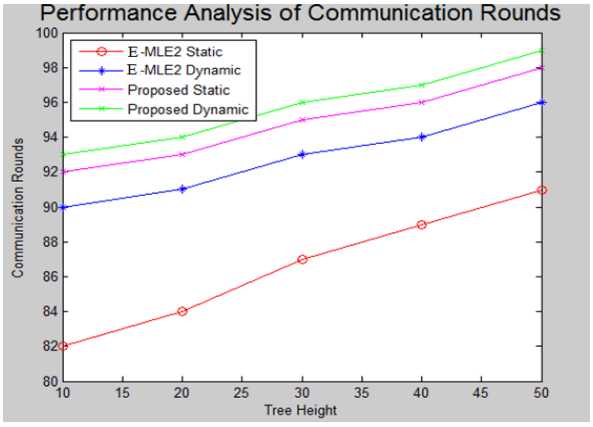

Fig.5. Performance Analysis of Communication Rounds

As observed from Fig.5, the average communication rounds obtained by the proposed method is 0.97, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So, the proposed method is considered as the best method. Fig.6 shows the communication rounds values with various file size of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

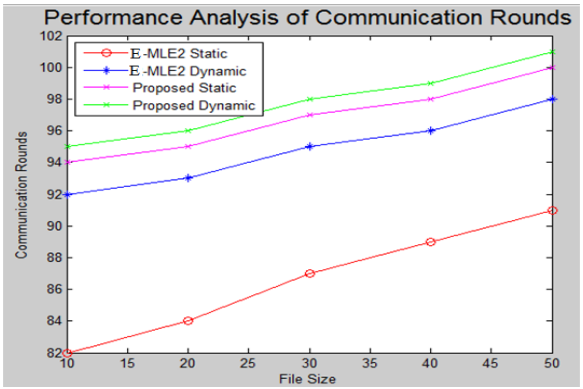

Fig.6. Performance Analysis of Communication Rounds Based on File Size

As observed from Fig.6, the average communication rounds obtained by the proposed method is 0.99, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So, the proposed method is considered as the best method.

4.3.3. Performance analysis using time taken value

In this experiment, we will evaluate the performance using the total time taken. The proposed method is compared with E-MLE2 Static and E-MLE2 dynamic methods. Ideally, a good data deduplication approach is expected to have a less time taken value. Fig.7 shows the time taken values for various file size of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

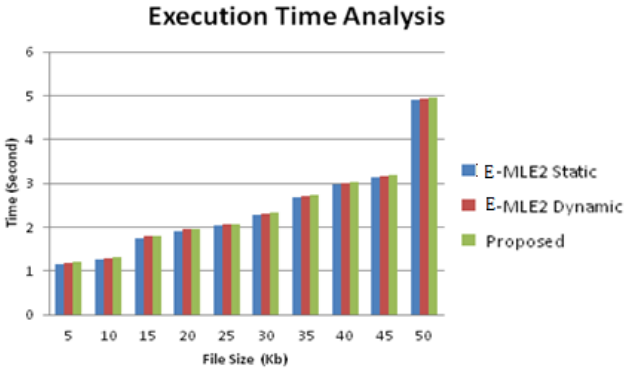

Fig.7. Performance Analysis of Execution Time Analysis

As observed from Fig.7, the average execution time obtained by the proposed method is slightly greater than the E-MLE2 Static and E-MLE2 dynamic methods. But based on the communication bits and rounds the proposed method performs best than the other two approaches. So, the proposed method is considered as the best method even though it provides less value.

4.3.4. Performance analysis using memory usage value

In this experiment, we will evaluate the performance of the data deduplication approach, using the memory space occupied. Ideally, a good data deduplication approach is expected to have a less occupied memory space. Fig.8 shows the memory taken values with various file size of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

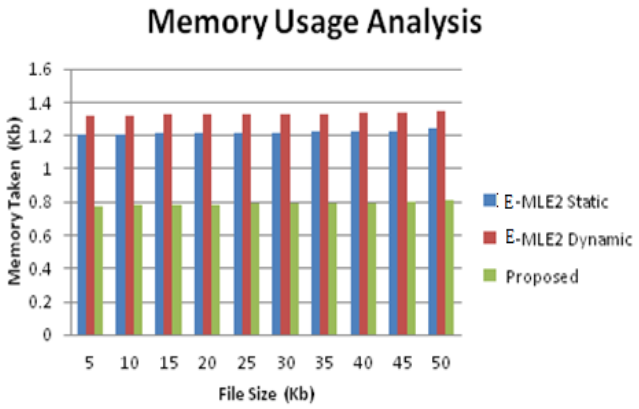

Fig.8. Performance Analysis of Memory Usage Analysis

As observed from Fig.8, the average memory taken value obtained by the proposed method is lower than the E-MLE2 Static and E-MLE2 dynamic methods.

4.3.5. Third party user malicious attempts and successful rate

Handling the malicious Third-Party User role is the main obstacle to data privacy protection. In a cloud storage system, users host their data on a server that is accessible from anywhere, called a cloud server. Data on the Cloud Server may be harmed by Third Party User fraud because of data outsourcing. Additionally, the private data of Cloud Users may be misused or disclosed to enemies. In Fig. 9, we contrast the Third-Party User's harmful attempts with the successful malicious detection made by the suggested static and dynamic deduplication approaches.

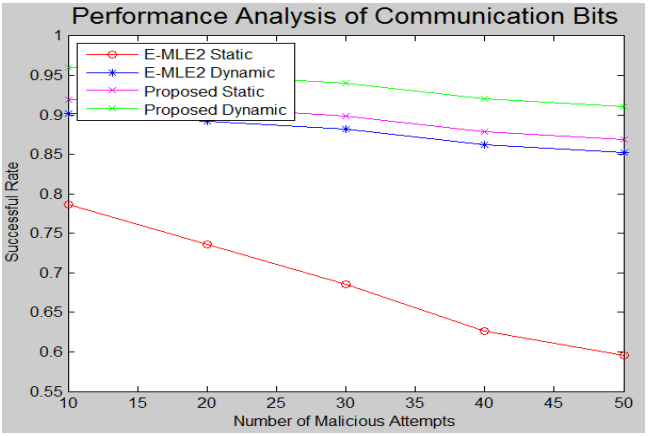

Fig.9. Performance Analysis of Malicious Attempt Analysis

4.3.6. Reliability of cloud server versus number of auditing

A major worry is the Cloud Server's dependability. The security of the customer's data in the data centre and the security of how the cloud services are supplied to the cloud users are the primary metrics used to assess the reliability of the cloud server. The dependability of the Cloud Server is shown in Fig. 10 for the E-MLE2 Static, E-MLE2 Dynamic, Proposed Static, and Proposed Dynamic deduplication techniques. Our experiment's main objective is to validate the Proposed Static and Proposed Dynamic and assess the Cloud Server's reliability in relation to the quantity of audits done on the client's data and cloud services.

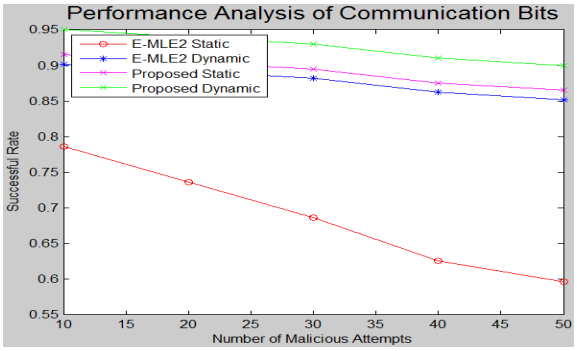

Fig.10. Performance Analysis of Reliability

4.3.7. Perfomance analysis of proposed system based on data duplication percentage

In this experiment, the proposed system is evaluated by using *communication* bits, communication rounds, time taken and memory usage under various values percentage of deduplication content in file. Fig.11 shows the communication bit values with various deduplication percentage of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

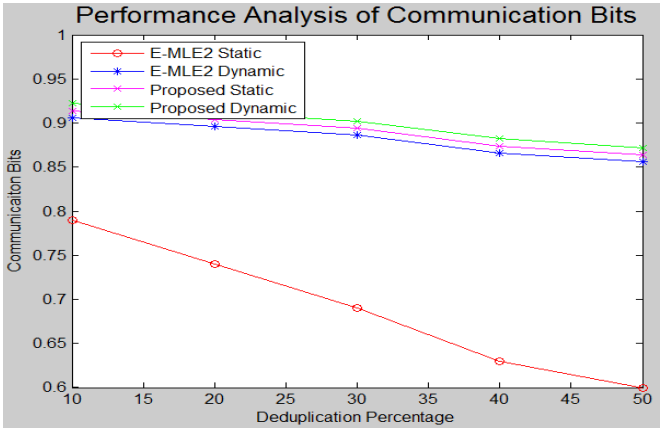

Fig.11. Performance Analysis of Communication Bits Based on Deduplication Percentage

As observed from Fig.11, the average communication bits obtained by the proposed method is 0.976, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So the proposed method is considered as the best method.

Fig.12 shows the communication rounds values with various deduplication percentage of Proposed Method, E-MLE2 Static and E-MLE2 dynamic methods.

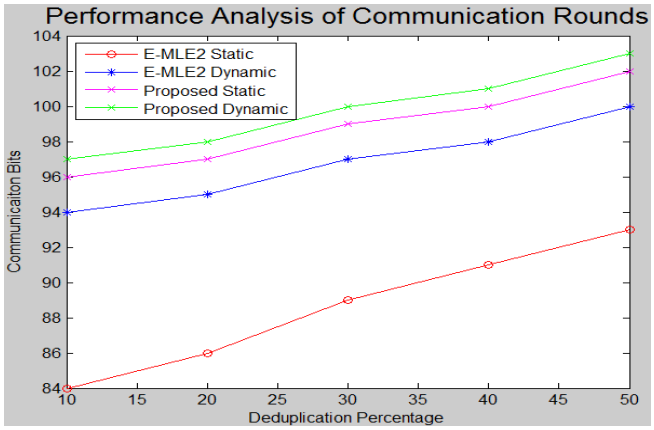

Fig.12. Performance Analysis of Communication Rounds Based on Deduplication Percentage

As observed from Fig.12, the average communication rounds obtained by the proposed method is 0.992, which is higher than that of the E-MLE2 Static and E-MLE2 dynamic methods. So the proposed method is considered as the best method.

5. CONCLUSION

This paper presents an enhanced method to improve the security of cloud-based deduplicated data. This method not only bolsters data reliability but also ensures the confidentiality of the data owner. The proposed technique involves the generation of security tags for data, which are established through a meticulous deduplication assessment for every data block. As a result, distinct tags are assigned to each unique data entry. The research incorporates two distinct schemes: the static scheme and the dynamic scheme. The static scheme is able to considerably reduce user effort while server does all process but in dynamic scheme allows the user to adjust the tree by increasing some computation cost. The static deduplication decision tree is built using the client's random elements, which prevents the tree from updating. The self-generation tree is used as the foundation for the dynamic deduplication decision tree, allowing the server to perform tree updates and other optimizations. The experimental analysis shows that the improved approach performs best than the existing approaches.

REFERENCES

[1] A. Arasu, M. Gotz, and R. Kaushik, "On active learning of record matching packages," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 783–794.
[2] A. Arasu, C. R_e, and D. Suciu, "Large-scale deduplication with constraints using dedupalog," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 952–963.
[3] R. J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all pairs similarity search," in Proc. 16th Int. Conf. World Wide Web, pp. 131–140, 2007.
[4] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi, "Active sampling for entity matching," in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 1131–1139.
[5] A. Beygelzimer, S. Dasgupta, and J. Langford, "Importance weighted active learning," in Proc. 26th Annu. Int. Conf. Mach. Learn., pp. 49–56, 2009.
[6] M. Bilenko and R. J. Mooney, "On evaluation and training-set construction for duplicate detection," in Proc. Workshop KDD, 2003, pp. 7–12.
[7] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," in Proc. 22nd Int. Conf. Data Eng., p. 5, Apr. 2006.
[8] P. Christen, "Automatic record linkage using seeded nearest neighbour and support vector machine classification," in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2008, pp. 151–159.
[9] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," IEEE Trans. Knowl. Data Eng., vol. 24, no. 9, pp. 1537–1555, Sep. 2012.
[10] P. Christen and T. Churches, "Febrl-freely extensible biomedical record linkage," Computer Science, Australian National University, Tech. Rep. TR-CS-02-05, 2002.
[11] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," Mach. Learn., vol. 15, no. 2, pp. 201–221, 1994.
[12] G. Dal Bianco, R. Galante, C. A. Heuser, and M. A. Gonalves, "Tuning large-scale deduplication with reduced effort," in Proc. 25th Int. Conf. Scientific Statist. Database Manage., 2013, pp. 1–12.
[13] M. G. de Carvalho, A. H. Laender, M. A. Goncalves, and A. S. Da Silva, "A genetic programming approach to record deduplication," IEEE Trans. Knowl. Data Eng., vol. 24, no. 3, pp. 399–412, Mar. 2012.
[14] A. Elmagarmid, P. Ipeirotis, and V. Verykios, "Duplicate record detection: A survey," IEEE Trans. Knowl. Data Eng., vol. 19, no. 1, pp. 1–16, Jan. 2007.
[15] I. Fellegi and A. Sunter, "A theory for record linkage," J. Am. Statist. Assoc., vol. 64, no. 328, pp. 1183–1210, 1969.
[16] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," Mach. Learn., vol. 28, no. 2-3, pp. 133–168, 1997.
[17] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in Proc. 3rd ACM Conf. Digital Libraries, 1998, pp. 89–98.
[18] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-off crowdsourcing for entity matching," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2014, pp. 601–612.
[19] H. Keopcke and E. Rahm, "Training selection for tuning entity matching," in Proc. Int. Workshop Quality Databases Manage. Uncertain Data, 2008, pp. 3–12.
[20] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2002, pp. 269–278.
[21] R. M. Silva, M. A. Goncalves, and A. Veloso, "A two-stage active learning method for learning to rank," J. Assoc. Inform. Sci. Technol., vol. 65, no. 1, pp. 109–128, 2014.
[22] S. Tejada, C. A. Knoblock, and S. Minton, "Learning domain-independent string transformation weights for high accuracy object identification," in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2002, pp. 350–359.
[23] R. Vernica, M. J. Carey, and C. Li, "Efficient parallel set-similarity joins using MapReduce," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 495–506.
[24] J. Wang, G. Li, and J. Fe, "Fast-join: An efficient method for fuzzy token matching, based string similarity join," in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 458–469.
[25] J. Wang, G. Li, J. X. Yu, and J. Feng, "Entity matching: How similar is similar," Proc. VLDB Endow., vol. 4, no. 10, pp. 622–633, Jul. 2011.
[26] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang, "Efficient similarity joins for near-duplicate detection," ACM Trans. Database Syst., vol. 36, no. 3, pp. 15:1–15:41, 2011.

The Authors have no conflict of interest to declare.

Authors Profile

Andal. V
Assistant Professor, MCA, M.Phil.
Graduated from Coimbatore has completed her MCA in Bharathidasan University, Tiruchirappalli, and MPhil. at Bharathiar University, Coimbatore. She has 23+ years of teaching experience in IT and related subjects as well as encouraging the student community through a variety of activities to succeed in their careers. Interested areas of research are Cloud computing and Cloud security.

.

Dr. D. Ganesh
Professor, MCA., M.Phil., MTech. in IT., Ph.D.
has completed his MCA in Bharathiar University Coimbatore, M.Phil. at M.S University, Tirunelveli, M.Tech. at Satyabhama University Chennai and Ph.D. in Image processing at Bharathiar University. He has 26+ years of experience in teaching and his professional excellence is through result-oriented approach, hard work, self-motivation and perseverance in teaching carrier. He has published papers in many Scopus journals, conferences and UGC care journals. His areas of research and teaching are Image Processing, Computer networks, Cryptography and Cloud security, Software Engineering, Machine Learning.