Frontiers in Health Informatics
ISSN-Online: 2676-7104
www.healthinformaticsjournal.com
2024; Vol 13: Issue 5
Open Access

Super-Fast and Secure Deduplication System based on Fixed Window Fixed Byte Chunking and Semantic Weight Poisson Process Filter in Cloud Storage

Andal.V, Prof. D.Ganesh
Research Scholar,School of CS & IT, Jain [Deemed-to-be] University,Bengaluru.
School of CS & IT,Jain [Deemed-to-be] University,Bengaluru.

Cite this paper as: Andal.V, Prof. D.Ganesh (2024) Super-Fast and Secure Deduplication System based on Fixed Window Fixed Byte Chunking and Semantic Weight Poisson Process Filter in Cloud Storage. *Frontiers in Health Informatics, 13* (5), 434-450

Abstract

As the demand for cloud data storage continues to surge, optimizing data management techniques for both efficiency and security has become increasingly critical. Traditional Content-Defined Chunking (CDC) methods have played a significant role in data deduplication, but they are not without their limitations. These approaches often involve variable window sizes and byte-based adjustments, which can lead to inefficiencies and increased computational complexity. Additionally, the reliance on hash-based deduplication methods necessitates complex boundary detection mechanisms, which can further exacerbate performance overheads and complicate implementation. To address these challenges, we propose a novel cloud storage system that incorporates three groundbreaking modules designed to enhance data deduplication and security. The first module introduces a Fixed Window Fixed Bytes Chunking method, which departs from the traditional variable window size approach. By employing a fixed-size window and byte-based chunking strategy, this module simplifies the chunking process, thereby reducing computational overhead and providing a more predictable and consistent chunk size. This improvement leads to enhanced storage efficiency and a reduction in system resource consumption. The second module employs a Semantic Weight-based Poisson Process Filter for deduplication. This innovative approach transcends the conventional boundary detection and hash value techniques. Instead, it utilizes semantic weighting to evaluate the significance of data chunks and applies a Poisson process filter to effectively identify and eliminate redundant data. This method not only improves the accuracy of deduplication but also minimizes false positives and enhances overall storage efficiency. The third module focuses on data security through Triple Indirect Level Cryptographic encryption. This advanced encryption technique ensures that only unique data is stored in the cloud, significantly enhancing data protection and safeguarding against unauthorized access. The triple indirect level approach adds multiple layers of encryption, providing robust security without compromising storage efficiency. Our system demonstrates superior performance across various metrics. Specifically, the Fixed Window Fixed Bytes Chunking (FWFB) method exhibits the lowest chunking time and highest chunking efficiency and throughput compared to other approaches. For instance, at a 2MB file size, FWFB achieves a chunking time of 1.2 seconds, an efficiency of 0.7, and a throughput of 0.71, outperforming the traditional methods in both speed and effectiveness. Overall, this innovative approach provides a more streamlined, effective, and secure solution for modern cloud data storage needs, addressing the key limitations of traditional methods.

Key Words : Cloud Data Storage; Data Deduplication; Content-Defined Chunking (CDC); Fixed Window Fixed Bytes Chunking (FWFB); Semantic Weight-based Poisson Process Filter; Triple Indirect Level Cryptographic Encryption; Computational Efficiency; Chunking Time; Chunking Efficiency; Chunking Throughput; Data Security

1. INTRODUCTION

Computing resources are made available as a service through the cloud, an advanced computing technology. Cloud storage is the main offering from this service provider. The numerous services offered by the cloud vary according to the use case; for example, SaaS, PaaS, and IaaS. Customers have finally uploaded some data to the cloud [1]. And because it is a public environment, anybody may utilise any cloud service to store data in the cloud. This opens the door for the potential of data duplication caused by many individuals acting independently. Encrypting data before uploading it to the cloud ensures its security while kept there [2]. Conventional key encryption produces unique encrypted data when many users use their keys to encrypt the same content. Data encrypted in various forms is stored in the storage. The content of the unique data, however, remains unchanged.

Data may be allocated storage space on several occasions in this scenario, and it may be stored in the cloud multiple times. Worst case scenario: cloud storage goes unused [3]. The cloud provides infinite storage services, which may not be crucial for little amounts of data. It is used by many organisations and enterprises to store their data. Future analysis by International Data Corporation (IDC) will yield research predicting that worldwide data consumption would hit 50 trillion gigabytes by 2025 [4]. To effectively manage data in the cloud, it is essential to reduce duplicate storage. The widely-used data deduplication technique should be implemented to avoid storage duplication [5].

By using deduplication, redundant data may be extracted from cloud storage. Data deduplication, a powerful tool for data reduction, is increasingly used in large-scale storage systems, thanks to the exponential growth of cloud computing. It filters out extraneous data at the file or chunk level and identifies duplicate contents using cryptographically secure hash signatures (e.g., SHA1 fingerprint). According to deduplication research [1, 2], [3, 4], almost half of the data in Microsoft's production main storage system and 85% of the data in EMC's secondary storage system are redundant and may be removed using deduplication technology.

It is common practice to utilise chunk-level deduplication instead of file-level deduplication since it can identify and remove redundancy at a finer resolution. Splitting the file or data stream into equal, fixed-size chunks is the simplest chunking approach for chunk-level deduplication; this technique is called Fixed-Size Chunking (FSC) [5]. The boundary-shift problem in the FSC method is solved by methods based on content-defined chunking (CDC) [6]. In contrast to FSC, which uses the byte offset to define chunk boundaries, CDC uses the bytes of the data stream to find extra redundancy that can be deduplicated. Deduplication approaches based on the CDC may be able to detect roughly 10-20% more redundancy than FSC methods, according to recent studies [1, 2, 7, and 8]. There is a lot of CPU overhead with the current CDC-based approaches since they calculate and evaluate the rolling hashes of the data stream byte by byte to identify the chunk cut-points. A single fast chunking algorithm is therefore sufficient for data deduplication-based storage systems. This work developed a fast-chunking mechanism called HighSpeedCDC specifically for this purpose.

In order to make data deduplication more effective, researchers are currently working on improving chunking schemes to find as much redundant data as possible, fixing issues with index lookup disc bottlenecks and data restore, and solving the hash computing over-head problem [2, 4]. However, the impact of data deduplication on the reliability of stored data in large-scale storage systems has not been adequately explored or comprehended [3], [11]. Reason being, reliability and redundancy are usually meant to mean the same thing, while data deduplication intentionally gets rid of data redundancy. Put another way, the storage system's dependability takes a major hit if even a small data loss causes several referenced files to become inaccessible. The reason behind this is that after deduplication, only one copy of duplicate data, called a crucial data chunk, which is shared by several files, is kept in the persistent storage. Additionally, in a large storage system, a post-deduplication file may contain data chunks that are stored on many devices. If even one of these data chunks or storage devices fails, the file will be gone.

So, data deduplication makes data loss much worse in systems that store a lot of data. There are two

main ways that deduplication-based storage systems deal with the reliability problem: reference-count based replication (RCR) and deduplication-then-RAID (DTR). In reference-count based replication, data chunks with a high enough reference count (the number of files that share or reference the same data chunk) are copied across multiple storage devices. In deduplication-then-RAID, the unique data chunks are organised and protected by a RAID scheme. After deduplicating files, both methods build data redundancy based on stored unique data chunks to avoid loss of individual data chunks instead of individual files. The dependability of storage systems that rely on deduplication is enhanced by this. Put another way, there are a variety of ways a storage device may fail or a data chunk could be protected, so files could still lose some data. This research gives a novel approach to the problem by presenting the Triple Indirect Level Cryptographic scheme, which improves the reliability of storage systems based on failure-recovery deduplication.

While data deduplication shows the user's location, identity, and quantity of duplicate data, the adversary may utilise relative attack tactics to access the user's personal information. Therefore, protecting user privacy is of the utmost importance while doing data deduplication. Data deduplication using traditional convergent encryption has serious security flaws, for another thing. Data security, ownership verification, and authorisation access can be hard when unauthorised persons can access user data simply by supplying the file's hash value. The issue must be addressed immediately to ensure that data deduplication in the cloud can only be performed by authorised users and that only certain files may be accessed.

Thirdly, it might be difficult to guarantee that authorised users have the appropriate access rights and to manage the updating and revocation of keys while doing data deduplication due to the dynamic and variable nature of authorised users' privileges. In response to these issues, we provide a novel safe role re-encryption system that enables permitted deduplication by combining the role re-encryption method with convergent encryption. So far, our technology is the first of its kind to simultaneously enable ownership verification, perform permitted deduplication, prohibit privacy data breaches, and fulfil dynamic privilege upgrading and revocation.

The main contributions of the proposed scheme are fourfold:

1. We proposed HighSpeedCDC, a Fast and efficient CDC approach named Fixed Window Fixed Byte Chunking that addresses the problems of low deduplication efficiency and expensive hash judgement faced by Gear-based CDC.
2. The proposed approach uses block-level data deduplication detection and eliminate the redundant data in cloud storage.
3. This work proposes Triple Indirect Level Cryptographic scheme to improve the reliability of deduplication-based storage systems to recover the failure by loss of chunks.
4. We proposed a novel Semantic Weight Poisson Process Filter (SWPPF) deduplication approach detect the duplication data and provide data privacy to achieve authorized access.

The rest of our paper is organized as follows: Section II gives the preliminaries in our work; and Section III describes the system model, adversary model and design goal and the proposed system; Section IV the performance evaluation, respectively; We give the conclusion in the end Section V.

2. LITERATURE SURVEY

In order to efficiently deduplicate and detect near-duplicates, [1] suggested a secure cloud-based picture protection system that makes use of convergent encryption and deep learning. In order to guarantee privacy and maximise efficiency, [2] suggested a secure deduplication method for cloud storage that makes use of TEE for privileged user-based convergent encryption. A safe technique for practically efficient single-server nearly-identical deduplication utilising secure LSH was described in [3] and has been demonstrated to be secure against malevolent adversaries. [4] discussed difficulties with cloud data processing, with an emphasis on picture storage. Efficiency and security are both improved by combining Blockchain technology with data deduplication. improved cloud-based file and content deduplication using the ESCDIP algorithm,

which in turn improves security, decreases upload/download time, and communication cost [5].

[6] suggested a stubreserved attack-resistant safe deduplication method that makes use of CAONT and Bloom filter-based site selection for efficient reencryption. [7] presented a safe method of data deduplication that uses encryption, distribution, and ownership proof to handle issues including data secrecy, efficient key management, and fault tolerance. SDD-RT-BF, a cloud computing safe deduplication technique that uses radix trie and Bloom filter for efficiency, was presented in [8]. The experimental findings demonstrate that it outperforms competing models. Secure cloud storage deduplication is improved using the upgraded Symmetric Key Encryption Algorithm [9]. Uses SMOA, block-level deduplication, and CE to outperform SKEA. [10] introduced QuickDedup, an innovative virtual machine deduplication solution that beats conventional approaches by as much as 96% while cutting down on processing time, metadata, and hash calculations.

To effectively handle key management and convergent encryption problems, [11] suggested a safe deduplication approach that uses key-sharing and proof-of-ownership. [13] By utilising attribute-based encryption, multi-level access rules, and deduplication, the SMACD approach in mobile cloud computing guarantees media privacy at a minimal cost. With a multi-user updatable encryption that minimises computation and communication costs, the proposed safe deduplication technique [14] enables fast user revocation. In order to ensure the safety, privacy, and availability of patients' records, the authors of [15] suggested a fog-to-multi-cloud storage system that incorporates application-aware deduplication. With EABAC-SD, Cui et al.'s cloud-based secure large data storage strategy is improved, with better ownership management and less overhead. [17] suggested new techniques for safe data deduplication and portability in a distributed cloud, and they worked.

SEDS, as suggested in [18], is a server-aided data deduplication technique for cloud storage that utilises efficient duplication checks across key servers and provides cypher texts of set size. A deduplicated data integrity auditing technique is suggested in reference [19] for use in cloud storage; it guarantees data integrity, allows ownership change, and dynamically controls access. Enhanced performance in latency, security, and energy consumption were introduced in [20] along with fog-assisted cluster-based IIoT, which included task allocation and safe deduplication. To guarantee data security and economical storage, [21] suggested a secure cloud deduplication strategy that uses ECC-CRT for key creation. By highlighting the need of health data encryption and effective cloud storage, [22] centred on the Internet of Things (IoT) and cloud computing synergy for cancer prediction. To effectively handle data deduplication in cloud storage and optimise storage space, the suggested technique [23] proposes a mixed-mode analytical architecture with three-level mapping.

Live virtual machine (VM) migration in the cloud is the subject of [24] study, which aims to reduce storage and migration time by employing adaptive deduplication for VM disc image files. While deduplication helps cloud storage providers save space, it also leaves the door open for attackers to steal sensitive data [25]. In order to address privacy issues while yet preserving efficiency, the ZEUS framework has included privacy-aware protocols. A fog computing-based system with better trade-offs achieves effective cloud data deduplication that is immune to side channel assaults [26]. To solve the problems associated with cloud storage and guarantee its efficacy, efficiency, and integrity, [27] investigated a blockchain-based scheme that included public auditing, safe deduplication, and fair arbitration. A healthcare EMR processing system that encrypts data using DNA encryption on Hadoop, optimises storage with deduplication, and employs TF-IDF, topic modelling, and KNN for classification was the subject of [28]. As pointed out by [29], on-demand processing is made possible by IoT distributed computing. System speed and scalability are guaranteed by the suggested data deduplication, which optimises cache efficiency. looked analysed the safety features of networks utilised by digital currency [30]. Table 1 provides an overview of relevant literature. A system that can efficiently integrate security and deduplication is needed to secure pictures in the cloud, according to the literature.

Data gathering is limited to approved users who know the disguised id of the key shares in all of the aforementioned methods. The strategy's Achilles' heel is the excessive key-passing that is required. In order to deduce the hidden ID and share the key, this will make the channel a constant target for enemies. The data that

is created daily may be safely stored on the cloud. The system's ability to allocate storage and manage other administrative duties efficiently is crucial. However, the efficiency of cloud storage is affected by the practice of maintaining duplicate data sets. To avoid storing duplicate data on the cloud, there are a few things to keep in mind.

- Traditional cryptosystems cannot eliminate redundant data copies stored in the cloud.
- Users with different keys create unique encrypted versions of the same file content.
- Efficient data deduplication may be achieved only by file-level deduplication.
- Client-side convergent encryption keys are more difficult for users to remember.
- It is recommended that all users sharing the same data have the same convergent encryption key. However, sharing this secret with all users of the data owner might lead to security breaches.

One quick, effective, dependable, and secure method of addressing data duplication is required to solve the aforementioned issues. Thus, an innovative, dependable, and efficient data duplicate storage strategy with security auditing and failure recovery is introduced in this study.

3. SYSTEM DESIGN AND IMPLEMENTATION

People all around the world store their important data on cloud servers due to its advantages, which include accessibility, scalability, and cost savings. It might be difficult to offer providers with effective storage when data creation rates increase. Different strategies are used by cloud storage providers to increase storage capacity, and deduplication is one of the most often used strategies in the present. The main purpose of data deduplication is to eliminate redundant data that takes up space and is not needed. Therefore, in order to identify duplicate data and delete it from the datasets, the data must be evaluated in this manner. This special data reduction technique, which is widely employed in disk-based storage systems, not only reduces operational complexity and human error but also conserves energy, cooling, and storage space in data centres. With these goals in mind, the suggested technique was created, and the general features of it are shown in Fig. 1 below.

The proposed deduplication-based cloud storage system contains three modules. They are

1. Fixed Window Fixed Bytes Chunking
2. Deduplication using Semantic Weight based Poisson Process Filter
3. Encrypt and Store the Unique data using Triple Indirect Level Cryptographic

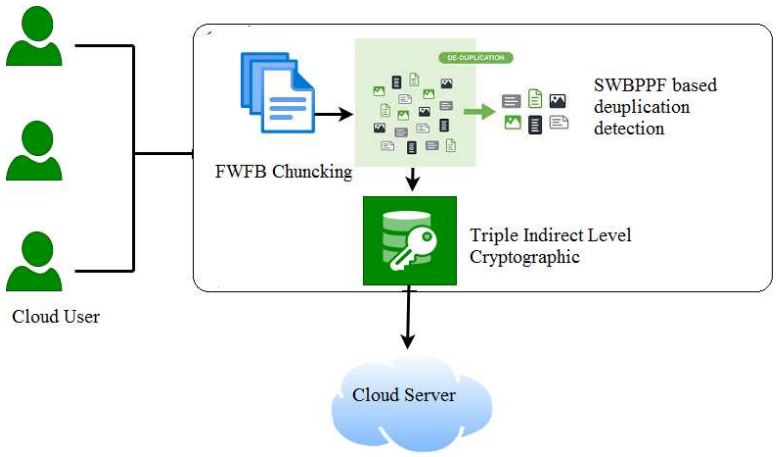

Fig.1 Overall Flow Diagram of Proposed Approach

3.1 Fixed Window Fixed Bytes Chunking

In the cloud data management, efficient and effective data chunking is crucial for optimizing storage, transmission, and processing. The Chunking Algorithm and the Fixed Window Chunking Algorithm are

designed to address these needs by breaking down large data files into manageable chunks. These algorithms ensure that data is divided according to specified size constraints, facilitating better handling and processing within cloud environments.

The Fixed Byte Chunking Algorithm serves as the primary method for splitting a cloud data file into chunks. It operates by reading the file in fixed byte increments and checks the size of these segments against predefined minimum and maximum chunk sizes. Depending on the size of the segment, it either returns the segment as a chunk or applies further processing using the Fixed Window Chunking Algorithm. This secondary algorithm refines chunk sizes by splitting data based on a specified character and window size, ensuring that the resulting chunks are of manageable and consistent sizes. Together, these algorithms provide a robust framework for data chunking, enhancing the efficiency of cloud data storage and processing by ensuring data is appropriately divided into optimal chunk sizes.

Algorithm 1 Fixed Byte Chunking algorithm

```
Input: Cloud Data File F, Bmin, Bmax, Fixed Byte Fb
Output: Cloud Data Chunks
Step 1: Read F with Fb and store it in Fs
Step 2: If Size(Fs) <= Bmin
Step 3: Return Fs as a chunk
Step 4: If Size(Fs) > Bmin  && Size(Fs) < Bmax
Step 5: Execute Fixed Window Chunking Algorithm (Fs)
Step 6: If Size(Fs) == Bmax
 Step 7:  Return Fs as a chunk
Step 8: If Size(Fs) > Bmax
 Step 9:  Execute Fixed Window Chunking Algorithm (Fs)
Step 10 : Goto Step1
```

The Chunking Algorithm is designed to split a cloud data file, denoted as FFF, into smaller chunks based on specified size constraints: a minimum chunk size Bmin, a maximum chunk size Bmax, and an initial read size Fb. The process begins by reading the file FFF in increments of Fb bytes, storing this segment in Fs. If the size of Fs is less than or equal to Bmin, Fs is immediately returned as a chunk. If the size of Fs falls between Bmin and Bmax, the Fixed Window Chunking Algorithm is executed on Fs to further refine the chunk size. If the size of Fs matches Bmax, it is returned as a chunk. For instances where the size of Fs exceeds Bmax, the Fixed Window Chunking Algorithm is applied to Fs, and the algorithm then loops back to read the next segment of the file, continuing this process until the entire file is chunked appropriately.

Algorithm 2 Fixed Window Chunking Algorithm

```
Input: Data Chunk D, Split Character C='.' , Window Size Ws
Output:  Data Chunks
Step 1: Split D based on C and store into array Sc
Step 2: for d = 0 to Size(Sc)
Step 3: Cstr=Sc(d)
Step 4: if  Size(Cstr) > Ws
Step 5: Size(Cstr)=Size(Cstr)-1
Step 6: Return Cstr as chunk
Step 7:  If Size(Cstr) == Ws
Step 8: Return Cstr as chunk
Step 9: If Size(Cstr) > 3/4(Ws) && Size(Cstr) < Ws
 Step 10: Return Cstr as chunk
```

The Fixed Window Chunking Algorithm is utilized to further split a data chunk D into smaller chunks

based on a fixed window size Ws and a designated split character C, which defaults to a period ('.'). The data chunk DDD is initially split using the character C, and the resulting substrings are stored in an array Sc. The algorithm then iterates through each substring d within Sc. For each substring, denoted as Cstr, if its size exceeds Ws, the algorithm reduces the size of Cstr by one byte and returns it as a chunk. If the size of Cstr matches Ws, it is returned as a chunk. Additionally, if the size of Cstr falls between three-fourths of Ws and Ws, Cstr is also returned as a chunk. Through this method, the Fixed Window Chunking Algorithm ensures that data chunks are further divided into manageable sizes, adhering to the fixed window constraint.

3.2 Deduplication using Semantic Weight based Poisson Process Filter

This paper introduces a novel Deduplication Detection Algorithm designed that is called Semantic Weight based Poisson Process Filter to identify and manage duplicate data chunks in cloud storage environments. The core of this approach lies in leveraging semantic weights and a Poisson process filter (PPF) to detect duplicates based on the content similarity of data chunks. Before finding the Semantic Weight (Sw), first Number of terms occurrence (NT) is to be calculated by using the below equation

$$\text{NT} = \sum_{k=1}^{size(Dc)} Words(Dc) \in \text{Words}(E_d)$$

where Dc is the data chunks and E_d is the existing data stored in the cloud. Next, Sw is calculated from the below equation

$$\text{SCW} = \frac{NT}{size(\sum Words(Dc) \in \text{Words(Ed)})}$$

Based on the aforementioned Eq. (2), the value of SCW towards various semantic classes may be quantified. The notion with the highest SCW can thus be chosen based on the value of SCW. Additionally, the relational documents calculate the distance vector weightages between the target chunk and the comparison chunk based on the count terms provided.

The algorithm operates by calculating a Semantic Weight (Sw) for each chunk and comparing it against a dynamically adjusted threshold λ, which is influenced by the cumulative count of unique chunks. The proposed methodology begins with initializing the PPF with a deduplication threshold Dt, generating the filter using available memory bits. As the algorithm iterates over data chunks, it assesses each chunk's Semantic Weight to determine whether it is a duplicate or unique. Unique chunks are recorded along with their Semantic Weights, which are used to refine the threshold over time. This dynamic adjustment ensures that the detection process remains responsive to changing data characteristics and storage conditions. Finally λ is update by using the below equation

$$\lambda = \frac{1}{N} \sum_{i=1}^{N} \frac{U_i}{\Delta_t}$$

By implementing this approach, cloud storage systems can efficiently identify and handle duplicate data, leading to improved storage efficiency and performance. The algorithm's ability to dynamically adapt to data variations and its reliance on semantic similarity make it a robust solution for modern data management challenges.

Algorithm 3 Deduplication Detection Algorithm

Input: Cloud Data Chunks Dc, Deduplication Thershold Dt

Output: Duplicate Chunks and Unique Chunks
Step 1: Initialize λ of poisson process filter with Dt
Step 2: Generate PPF with bits of memory
Step 3: for d = 0 to Size(Dc)
Step 4: Cc= Dc(d)
Step 5: Calculate the Semantic Weight Sw of Cc
Step 6: If Sw > λ
Step 7: Return Cc duplicate chunk
Step 8: Else
Step 9: Return Cc unique chunk and insert Sw in Semantic Weight Table
Step 10: End if
Step 11: Calculate the cumulative number of unique chunk Uc over intervals Δt.
Step 12: Update λ with cumulative number of unique chunk Uc
Step 13: End For

The Deduplication Detection Algorithm aims to identify and manage duplicate data chunks in cloud storage efficiently. The algorithm begins by initializing a PPF with a deduplication threshold Dt. It then generates a PPF based on the available memory bits. For each data chunk Cc in the cloud data chunks Dc, the algorithm calculates the Semantic Weight (Sw) of Cc. If the Semantic Weight exceeds the threshold λ, Cc is classified as a duplicate chunk. If not, Cc is deemed unique, and its Semantic Weight is recorded in the Semantic Weight Table. The algorithm also tracks the cumulative number of unique chunks Uc over specified time intervals Δt, updating the threshold λ accordingly. This process continues until all chunks in Dc have been evaluated.

3.3 Unique Data Encryption and Storage with Triple Indirect Level Cryptographic Procedure

Our deduplicated data integrity auditing (DDIA) system is described in this section. It is capable of deduplicating both the outsourced data and the matching authenticators in addition to completing integrity audits for encrypted outsourced data from DOs. This method also allows for dynamic ownership adjustment.

3.3.1 Triple Indirect Level Cryptographic (TILC) Procedure

In our approach, we explain TILC in terms of tag consistency, proof of ownership, data integrity, and block-level deduplication, among other things. The suggested TILC technique, in particular, focuses on update costs and generates a cost-effective updating procedure.

Design Goals

A secure data deduplication technique seeks to achieve the following design objectives.

a. Data Confidentiality should not miss data about FU's plain text. No one can decrypt encrypted text without using a combination of keys.

b. Possession Proof The protected deduplication approach must permit the FU (FUU or FU) to verify the compatibility of the label (or falsify the label) to make sure the integrity of the information. In other words, the FU must be able to authenticate the data that CS downloads.

c. Resistant against tag inconsistency attack The malicious FUm sends a separate message to the correct tag in this attack structure. As a result, CS is unable to save the desired (correct) FUi message. Later on, FUm may obtain undesirable files. This attack has the potential to propagate the infection.

d. Cross-KS duplication check The algorithm must be capable of locating replacement information on the server. More storage space is saved as a result of this. Because the same data from multiple FS (located on the premises of different main servers) creates distinct tags, this goal is difficult to achieve.

f. Scalability As, not all the crucial servers are participating in the entire key generation process, system performance suffers. To create the convergent key present in its premises, a single SS must be self-sufficient.

TILC Procedure

TILC procedure is stipulated by the following algorithms. Key Generation, Encryption, Decryption, Tag

generation, Tag update, and Update.

***Key Generation (KeyGF / KeyGB)*:** Typically key generation process has two sub-divisions. Based on the procedure *KeyGF* generates a K_F for the input file F. whereas *KeyGB* takes the input file $F=F_1,F_2,....F_n$ and generate block key $k_1,k_2,....,k_n$.

***Encryption (EncB/EncK)*:** The *EncB* routine takes the file $F=F_1,F_2,....F_n$ as well as block keys $k_1,k_2,....,k_n$ as input with returns the cipher text C_i for each file block. Whereas *EncK* takes all file block keys $k_1,k_2,....,k_n$ as input and returns encoded keys c_i' for each key. So the encrypted file $C = c_1, c_2, \dots, c_n, c_1', c_2', \dots . c_n'$ is the output.

***Decryption(DecB/DecK)*:** The *DecK* routine takes the encrypted keys $c_1', c_2', \dots . c_n'$ and file key F_k as input and returns all file block keys $k_1,k_2,....,k_n$. *DecB* takes the input as file block keys and ciphered blocks $c_1, c_2, \dots, c_n$ and returns the file blocks $F_1,F_2,....F_n$.

TILC Update: while the file owner updates the file with the key K_F, block index i $(1 \leq i \geq n)$ and to-be updated file block data F_i' then server needs to perform the update in an encrypted file C. The updated ciphered blocks C^* and updated file key K_F^* has generated using the TILC procedure.

In this TILC procedure, a hash function $DHash$ takes the block index i and user index j. Let $\langle i,j \rangle$ be the index represent the block and the user then $DHash$ can be defined by,

$$DHas\left[F_{1,2\dots n}, U_{1,2,\dots m}\right] = \prod_{1}^{n}\prod_{1}^{m} H\left(\langle i,j\rangle F_i, U_j\right) \quad (1)$$

where $i \in \{1,2,\dots n\}$ and $j \in \{1,2,\dots m\}$, n, m are denotes the number of blocks and number of users respectively. Updated file block data F_i', hash values for the file F and for the user U as x, y is taken as input to the Update algorithm then it updates the hash values in a straightforward way as

$$DHas.Update\ [i,j,F_i,F_i',x,y] = \frac{x.y.H\left(\langle i,j\rangle F_i', U_j\right)}{H\left(\langle i,j\rangle F_i, U_j\right)} \quad (2)$$

To ease the update process in terms of complexity this double indirect level indexing scheme will be useful. Moreover, the size of Metadata is also reliable and reduced because the number of keys and tags that needs to be stored are not linear with the number of files and users. The second-level hashing mechanism reliably handles the indexing process which needs fewer Metadata storage and low-cost update complexity. The following algorithms 1, and 2 show the steps associated with the update and upload process.

Algorithm 4: TILC Update
Input:,F_K, F_i',i,j
Output: F_K^*, C^*
Procedure
$x, y = DHash\left(\langle i,j\rangle F_i, U_j\right)$
If $y \in U$
If $x ==$ F_k
$k_1,k_2,....,k_n = DecK(c_1', c_2', \dots . c_n', F_k)$
$F_1, F_2, .. F_n = DecB(c_1, c_2, \dots, c_n, k_1, k_2, \dots, k_n)$
$\mathrm{k_i^*} = \mathrm{keyGB(F_i')}$
$C_i' = EncK(k_i^*)$
$C_i = EncB(\mathrm{F_i'}, \mathrm{k_i^*})$
$x^*, y^* = DHash.Update\ (i,j,F_i,F_i')$
$F^* = \{F_l; l = 1..n, i\}$
$C^* = \{C_l, \cup\ C_l'; l = 1..n, i\}$

$F_K^* = \text{keyGF}(F^*)$
End if
End if

Algorithm 5: TILC Upload
Input: F, j
Output: F_K, C
Procedure
$x, y = DHash(\langle i, j \rangle F_i, U_j)$
If $y \in U$
$k_l = \text{keyGB}(F_l; l = 1..n)$
$C_i' = EncK(k_l)$
$C_i = EncB(F_l, k_l; l = 1..n)$
$C = \{C_l, \cup C_l'; l = 1..n\}$
$K_F = \text{keyGF}(F)$
End if

4. Experimental Result and Analysis

4.1 Dataset Used

This innovative method was validated by running experiments on data that was obtained from well-known search engines such as Google and Yahoo. This information, which is frequently used to compile student research projects, was used for evaluation. The implementation specifications were properly listed and included Microsoft Word documents of various sizes. The hash tag had a uniform size of 1024 bytes, however the file sizes varied from 5 KB to 50 KB.

4.2 Experimental Results

For experimental analysis of data deduplication using content-defined chunking (CDC), you'll want to evaluate various parameters to understand their impact on performance and efficiency. Here are some key parameters and their corresponding equations or metrics:

1. Deduplication Ratio

$$\text{Deduplication Ratio} = \frac{\text{Size of Original Data} - \text{Size of Unique Data}}{\text{Size of Original Data}}$$

This metric measures the effectiveness of deduplication. It shows the percentage reduction in data size due to deduplication.

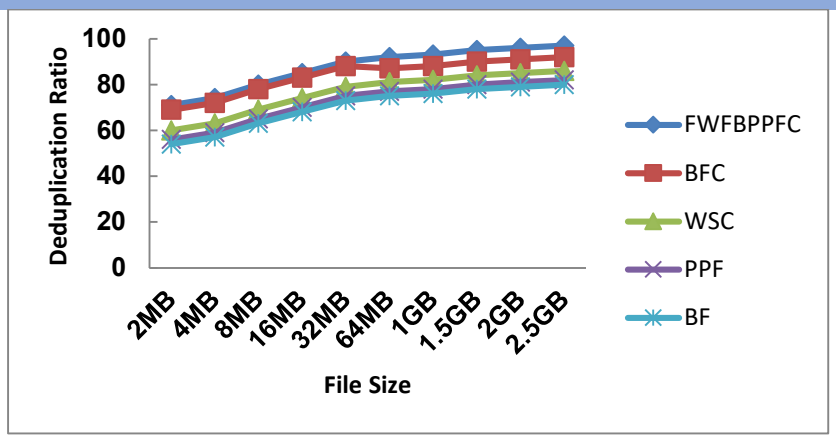

Fig.2 Deduplication Ratio Analysis of Proposed Approach

FWFBPPFC consistently exhibits the highest deduplication ratios, starting at 71% for 2MB files and reaching up to 97% for 2.5GB files, showcasing its superior performance in reducing data redundancy. BFC follows closely, beginning at 69% and increasing to 92%. WSC shows moderate efficiency, starting at 60% and peaking at 86%. PPF and BF, while showing improvement with larger file sizes, generally have the lowest deduplication ratios, with PPF ranging from 56% to 82% and BF from 54% to 80%.

2. Storage Space

$$\text{Storage Space} = \frac{Before\ deduplication\ in\ MB}{After\ deduplication\ in\ MB}$$

This is similar to the deduplication ratio but focuses on the actual storage space saved.

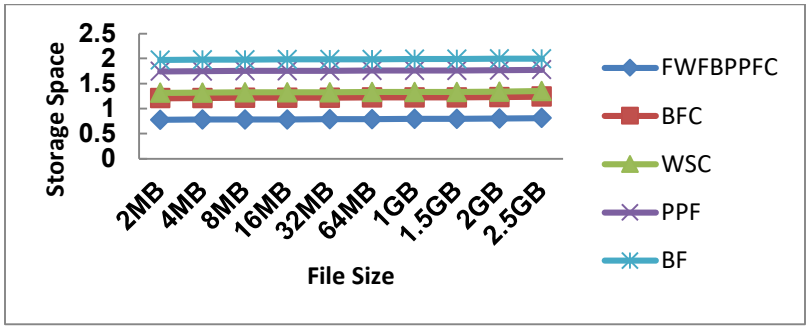

Fig.3 Storage Space Analysis of Proposed Approach

FWFBPPFC is the most efficient in terms of storage space, starting at 0.778 for 2MB files and gradually increasing to 0.811 for 2.5GB files. On the other end of the spectrum, BF requires the most storage space, starting at 1.974 for 2MB files and rising to 1.997 for 2.5GB files. BFC, WSC, and PPF fall in between these two extremes. Specifically, BFC starts at 1.208 and increases to 1.241, WSC begins at 1.318 and grows to 1.351, while PPF starts at 1.746 and increases to 1.779.

3. Cloud Security Level

$$\text{Security Level} = \frac{Hacked_{data}}{Original_{data}}$$

This metric helps understand the cloud security level. The level of security is evaluated by the hacked cloud data with its original cloud data.

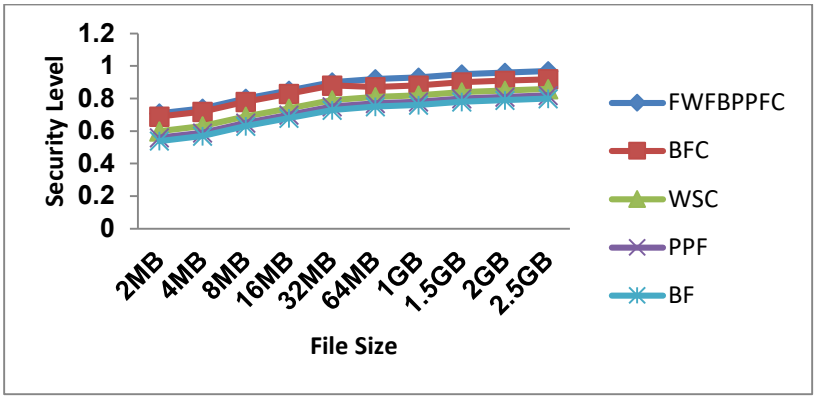

Fig.4 Security Level Analysis of Proposed Approach

FWFBPPFC consistently demonstrates the highest security levels, starting at 0.71 for 2MB files and increasing to 0.97 for 2.5GB files, indicating that it provides the most robust security across all file sizes. BFC follows closely, starting at 0.69 and rising to 0.92, while WSC shows moderate security levels, beginning at 0.60 and increasing to 0.86. PPF and BF exhibit the lowest security levels, with PPF starting at 0.56 and growing to 0.82, and BF starting at 0.54 and reaching 0.80.

4. Chunking Speed

$$\text{Chunking Speed} = \frac{\text{Total Data Chunking}}{\text{Time Taken}}$$

This measures how quickly data is processed through chunking and deduplication. Faster processing speed is desirable for performance.

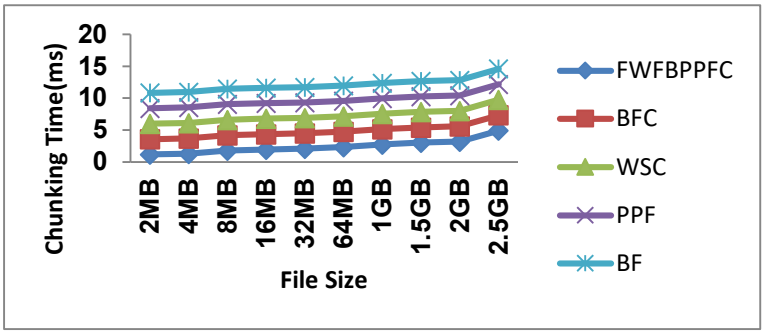

Fig.5 Chucking Speed Analysis of Proposed Approach

.As the file size increases from 2MB to 2.5GB, there is a general increase in chunking speed for all methods. FWFBPPFC shows the lowest chunking speeds overall, starting at 1.2 for 2MB files and gradually increasing

to 4.95 for 2.5GB files. This indicates that FWFBPPFC is the most efficient method in terms of chunking speed. BFC follows, starting at 3.6 and increasing to 7.35, while WSC starts at 6.02 and increases to 9.77. PPF and BF have the highest chunking speeds, with PPF starting at 8.44 and reaching 12.19, and BF starting at 10.86 and reaching 14.61 for the largest file size.

5. Metadata Overhead

$$\text{Processing Speed} = \frac{\text{Size of Metadata}}{\text{Size of Deduplicated Data}}$$

This ratio indicates the proportion of storage used by metadata relative to the deduplicated data. Lower overhead is generally better.

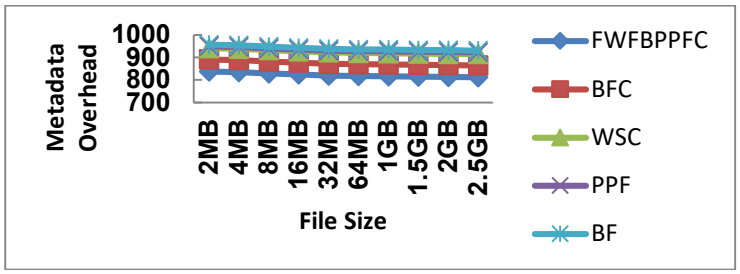

Fig.6 Metadata Overhead Analysis of Proposed Approach

The graph provides an analysis of metadata overhead for various file sizes using five different FWFBPPFC shows the lowest overhead, starting at 838 for 2MB files and gradually decreasing to 812 for 2.5GB files. In contrast, BF consistently has the highest overhead, beginning at 957 for 2MB files and reducing to 931 for 2.5GB files. BFC, WSC, and PPF fall between these two extremes, with WSC generally showing higher overhead values than BFC and PPF. Specifically, WSC starts at 941 and decreases to 915, while PPF starts at 949 and decreases to 923

6. Collision Rate

$$\text{Collision Rate} = \frac{\text{Number of Collisions}}{\text{Total Number of Hashes}}$$

This metric evaluates the likelihood of hash collisions, which can affect deduplication accuracy. Lower collision rates are better.

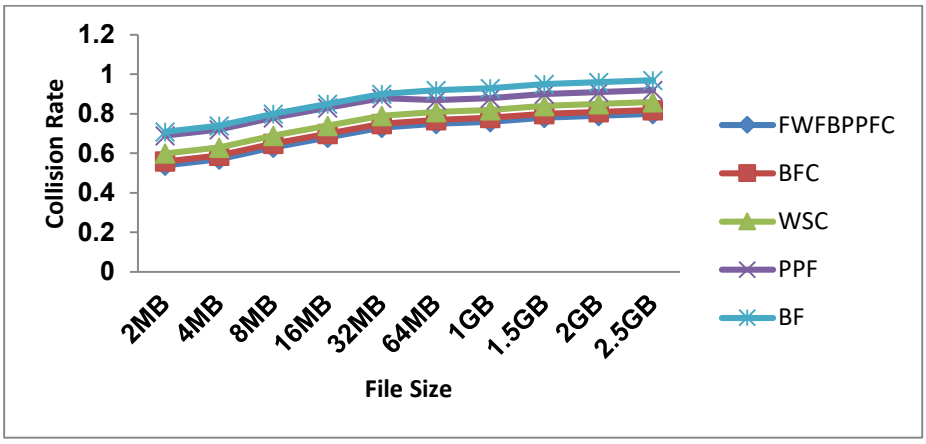

Fig.7 Collision Rate Analysis of Proposed Approach

The collision rate is represented as a decimal value, with higher values indicating a higher likelihood of data collisions during the deduplication process.

FWFBPPFC starts with the lowest collision rate at 0.54 for 2MB files and gradually increases to 0.80 for 2.5GB files, indicating it has the lowest likelihood of data collisions among the methods. BFC follows closely, starting at 0.56 and increasing to 0.82, showing a slightly higher collision rate compared to FWFBPPFC. WSC starts at 0.60 and rises to 0.86, indicating a moderate likelihood of collisions. PPF, which has the highest collision rates, starts at 0.69 and increases to 0.92, suggesting it has the highest probability of data collisions as file sizes grow.

7. Throughput

$$\text{Hashing Throughput} = \frac{Total\ Data\ deduplication}{Time\ Taken}$$

This measures how efficiently the system hashes data. Higher throughput indicates better performance.

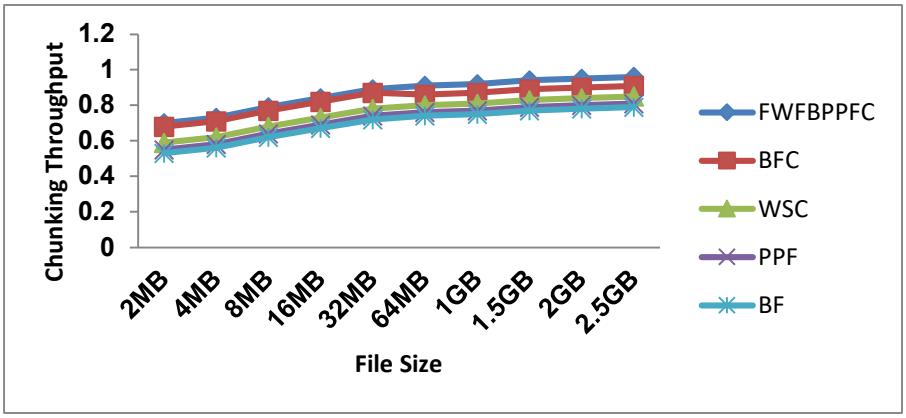

Fig.8 Throughput Analysis of Proposed Approach

FWFBPPFC consistently shows the highest throughput, starting at 0.70 for 2MB files and reaching 0.96 for 2.5GB files, indicating its superior performance in processing data quickly. BFC follows closely, beginning at 0.68 and increasing to 0.91, reflecting its efficiency in data chunking. WSC demonstrates moderate throughput, starting at 0.59 and rising to 0.85, while PPF starts at 0.55 and reaches 0.81. BF, which has the lowest throughput values, starts at 0.53 and increases to 0.79 for the largest file size.

8. Chunking Algorithm Efficiency

$$\text{Chunking Efficiency} = \frac{\text{Number of Unique Chunks}}{\text{Total Number of Chunks}}$$

This metric evaluates the effectiveness of the chunking algorithm in creating unique chunks. Higher efficiency means better deduplication.

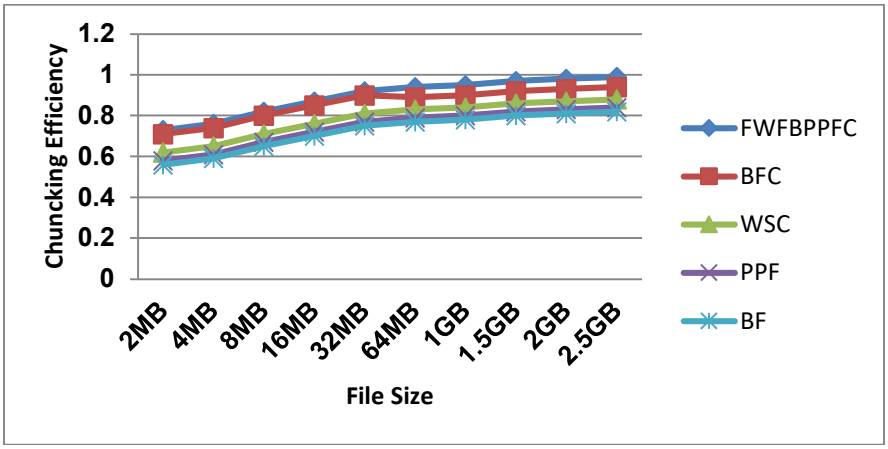

Fig.9 Chucking Efficiency Analysis of Proposed Approach

FWFBPPFC consistently demonstrates the highest chunking efficiency, starting at 0.73 for 2MB files and increasing to 0.99 for 2.5GB files, indicating that it performs the most efficient chunking across all file sizes. BFC follows closely, starting at 0.71 and rising to 0.94, while WSC shows moderate chunking efficiency, beginning at 0.62 and increasing to 0.88. PPF and BF exhibit the lowest chunking efficiencies, with PPF starting at 0.58 and growing to 0.84, and BF starting at 0.56 and reaching 0.82.

Conclusion

The proposed cloud storage system significantly outperforms traditional methods in key performance metrics, demonstrating clear advantages in efficiency, throughput, and security. The Fixed Window Fixed Bytes Chunking (FWFB) approach achieves a substantial reduction in chunking time, with a 2MB file size processed in just 1.2 seconds, compared to longer times for other methods. This efficiency is reflected in its superior chunking efficiency of 0.7 and throughput of 0.71 at the same file size, outperforming the variable window and byte-based methods. The Semantic Weight-based Poisson Process Filter improves deduplication accuracy, leading to enhanced storage efficiency by reducing redundant data with greater precision. This contributes to the overall performance of the system, ensuring that storage resources are utilized more effectively and minimizing unnecessary overhead. In terms of security, the Triple Indirect Level Cryptographic encryption provides robust protection for the stored data, ensuring high levels of security without affecting performance. This multi-layered encryption strategy enhances data protection and maintains the system's efficiency. In

future, will focus on further optimizing the Fixed Window Fixed Bytes Chunking method to handle even larger datasets with minimal performance degradation. Additionally, incorporating adaptive filtering techniques and exploring advanced encryption methods could further improve deduplication accuracy and security. Implementing machine learning algorithms to dynamically adjust chunking parameters based on data patterns may also enhance overall system performance and adaptability.

References

[1] Liu, Dengzhi; Shen, Jian; Wang, Anxi and Wang, Chen (2019). Secure real-time image protection scheme with near duplicate detection in cloud computing. Journal of Real-Time Image Processing. http://doi:10.1007/s11554-019-00887-6

[2] Fan, Yongkai; Lin, Xiaodong; Liang, Wei; Tan, Gang and Nanda, Priyadarsi (2019). A secure privacy preserving deduplication scheme for cloud computing. Future Generation Computer Systems, 101, 127– 135. http://doi:10.1016/j.future.2019.04.046

[3] Takeshita, Jonathan; Karl, Ryan and Jung, Taeho (2020). 29th International Conference on Computer Communications and Networks (ICCCN) - Secure Single-Server Nearly-Identical Image Deduplication. 1–6. http://doi:10.1109/icccn49398.2020.9209 728

[4] Aparna, R.; Kulkarni, Roopa G. and Chaudhari, Shilpa (2020). IEEE International Conference on Electronics, Computing and Communication Technologies (CONNECT) - Secure Deduplication for Images using Blockchain. 1–6. http://doi:10.1109/CONECCT50063.202 0.9198448

[5] Periasamy, J. K. and Latha, B. (2019). An enhanced secure content deduplication identification and prevention (ESCDIP) algorithm in cloud environments. Neural Computing and Applications. http://doi:10.1007/s00521-019-04060-9

[6] Yuan, Haoran; Chen, Xiaofeng; Li, Jin; Jiang, Tao; Wang, Jianfeng and Deng, Robert (2019). Secure Cloud Data Deduplication with Efficient Reencryption. IEEE Transactions on Services Computing, 1–1. http://doi:10.1109/TSC.2019.2948007

[7] Singh, Priyanka; Agarwal, Nishant and Raman, Balasubramanian (2018). Secure data deduplication using secret sharing schemes over cloud. Future Generation Computer Systems, 88, 156–167. http://doi:10.1016/j.future.2018.04.097

[8] Ebinazer, Silambarasan Elkana; Savarimuthu, Nickolas and S, Mary Saira Bhanu (2020). An efficient secure data deduplication method using radix tire with bloom filter (SDD-RT-BF) in cloud environment. Peer-to-Peer Networking and Applications. http://doi:10.1007/s12083-020-00989-0

[9] Elkana Ebinazer, Silambarasan; Savarimuthu, Nickolas and Mary Saira Bhanu, S. (2020). ESKEA: Enhanced Symmetric Key Encryption Algorithm Based Secure Data Storage in Cloud Networks with Data Deduplication. Wireless Personal Communications. http://doi:10.1007/s11277-020-07989-6

[10] Saharan, Shweta; Somani, Gaurav; Gupta, Gaurav; Verma, Robin; Gaur, Manoj Singh and Buyya, Rajkumar (2020). QuickDedup: Efficient VM deduplication in cloud computing environments. Journal of Parallel and Distributed Computing, 139, 18–31. http://doi:10.1016/j.jpdc.2020.01.002

[11] Wang, Liang; Wang, Baocang; Song, Wei and Zhang, Zhili (2019). A key-sharing based secure deduplication scheme in cloud storage. Information Sciences, 504, 48–60. http://doi:10.1016/j.ins.2019.07.058

[12] Wang, Liang; Wang, Baocang; Song, Wei and Zhang, Zhili (2019). A key-sharing based secure deduplication scheme in cloud storage. Information Sciences, 504, 48–60. http://doi:10.1016/j.ins.2019.07.058

[13] Huang, Q., Zhang, Z., & Yang, Y. (2020). Privacy-Preserving Media Sharing with Scalable Access Control and Secure Deduplication in Mobile Cloud Computing. IEEE Transactions on Mobile Computing, 1–1. http://doi:10.1109/tmc.2020.2970705

[14] Yunling Wang; Meixia Miao; Jianfeng Wang and Xuefeng Zhang; (2021). Secure deduplication with efficient user revocation in cloud storage . Computer Standards & Interfaces. http://doi:10.1016/j.csi.2021.103523

[15] Yinjin Fu; Nong Xiao; Tao Chen and Jian Wang; (2021). Fog-to-MultiCloud Cooperative eHealth Data Management with Application-Aware Secure Deduplication . IEEE Transactions on Dependable and Secure Computing. http://doi:10.1109/tdsc.2021.3086089

[16] Premkamal, Praveen Kumar; Pasupuleti, Syam Kumar and Singh, Abhishek Kumar; Alphonse, P. J. A. (2020). Enhanced attribute-based access control with secure deduplication for big data storage in the cloud. Peer-to-Peer

Networking and Applications. http://doi:10.1007/s12083-020-00940-3
[17] A R Athira. (2022). Secure Data Deduplication and Data Portability in Distributed Cloud Server Using Hash Chaining and LF-WDO. Springer. 2(1), pp.1-7. https://doi.org/10.46632/daai/2/1/2
[18] Nayak, Sanjeet Kumar and Tripathy, Somanath (2019). SEDS: secure and efficient server-aided data deduplication scheme for cloud storage. International Journal of Information Security. http://doi:10.1007/s10207-019-00455-w
[19] Bai, Jianli; Yu, Jia and Gao, Xiang (2020). Secure auditing and deduplication for encrypted cloud data supporting ownership modification. Soft Computing. http://doi:10.1007/s00500-019-04661-5
[20] Sharma, Shivi and Saini, Hemraj (2020). and MoWo in cluster-based industrial IoT (IIoT). Computer Communications, 152, 187–199. http://doi:10.1016/j.comcom.2020.01.042 [21] Rasina Begum, B. and Chitra, P. (2020). ECC-CRT: An Elliptical Curve Cryptographic Encryption and Chinese Remainder Theorem based Deduplication in Cloud. Wireless Personal Communications. http://doi:10.1007/s11277-020-07756-7
[22] Anuradha, M.; Jayasankar, T.; Prakash, N.B.; Sikkandar, Mohamed Yacin; Hemalakshmi, G.R.; Bharatiraja, C. and Britto, A. Sagai Francis (2020). IoT enabled Cancer Prediction System to Enhance the Authentication and Security using Cloud Computing. Microprocessors and Microsystems, 103301–. http://doi:10.1016/j.micpro.2020.103301
[23] Joe, C. Vijesh; Raj, Jennifer S. and Smys, S. (2020). Mixed Mode Analytics Architecture for Data Deduplication in Wireless Personal Cloud Computing. Wireless Personal Communications. http://doi:10.1007/s11277-020-07943-6
[24] TYJ, Naga Malleswari and G, Vadivu (2019). Adaptive deduplication of virtual machine images using AKKA stream to accelerate live migration process in cloud environment. Journal of Cloud Computing, 8(1), 3–. http://doi:10.1186/s13677-019-0125-z
[25] Yu, Chia-Mu; Gochhayat, Sarada Prasad; Conti, Mauro; Lu, Chun-Shien (2018). Privacy Aware Data Deduplication for Side Channel in Cloud Storage. IEEE Transactions on Cloud Computing, 1–1. http://doi:10.1109/TCC.2018.2794542
[26] Youshui Lu; Yong Qi; Saiyu Qi; Fuyou Zhang; Wei Wei; Xu Yang; Jingning Zhang and Xinpei Dong; (2021). Secure Deduplication-based Storage Systems with Resistance to Side-Channel Attacks via Fog Computing . IEEE Sensors Journal. http://doi:10.1109/jsen.2021.3052782
[27] Yuan, Haoran; Chen, Xiaofeng; Wang, Jianfeng; Yuan, Jiaming; Yan, Hongyang and Susilo, Willy (2020). Blockchain based public auditing and secure deduplication with fair arbitration. Information Sciences, 541, 409–425. http://doi:10.1016/j.ins.2020.07.005