Cluster Computing
https://doi.org/10.1007/s10586-023-04171-y

Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach

Mohd Akbar[1] · Irshad Ahmad[1] · Mohsina Mirza[2] · Manavver Ali[3] · Praveen Barmavatu[4]

Received: 8 May 2023 / Revised: 3 October 2023 / Accepted: 5 October 2023

Abstract
Enormous information is an accumulation of vast measures of information that contains copied things gathered from different sources. Distinguishing copied information from enormous information is a difficult issue. In this paper secure deduplication scheme has been presented for huge information stockpiling. Customers may store and transmit data effectively and efficiently due to cloud service providers. Providers of cloud services are drawn to the application of data de-duplication tools to lower storage costs and conserve bandwidth. To secure the data they share on the cloud, users of the cloud are interested in having a secure and private connection. As a result, the priory is transferred to the cloud, and the data is encrypted. Because the encryption objective conflicts with the purpose of data deduplication, this feature presents a difficult problem. The author then suggested brand-new hybrid chunking methods to boost deduplication performance in cloud storage. Content-based hybrid chunking algorithm of Two Threshold Two Divisor (TTTD) with a Dynamic Prime Coding (DPC) algorithm is suggested for reducing the deduplication data. To increase throughput, which means reducing execution time and CPU resource utilisation, an index method must be incorporated into the deduplication matching process. For chunks with higher security levels, it offers greater security protection, while for chunks with lower security levels, it trades off security for greater deduplication effectiveness. When moving data to open, semi-trusted systems like public clouds, identity management, secure data exchange, and privacy preservation become more challenging than they are in closed, trusted systems where security and privacy can be controlled more simply. Accordingly, the article proposed a cross-domain authentication technique for the verification process. Evaluations of the real-world datasets show the efficiency of the design. With other methods, the proposed hybrid Content-based TTTD-DPC analyses many performance parameters, including chunk distribution, processing time, chunking time, throughput, average chunk size, and deduplication ratio. The deduplication system's total throughput and processing speed are both enhanced by the proposed approach. In addition, the recommended methodology reduces computational cost (772 ms) in terms of hash function compared to other existing methods.

Keywords Deduplication · Chunking · Security system · Two threshold two divisor · Dynamic prime coding · Chameleon hash algorithm · Blacklist Merkel tree · Authentication · Machine learning

✉ Praveen Barmavatu
pbarmavatu@utem.cl

[1] Department of Information Technology, University of Technology and Applied Sciences, Muscat, Sultanate of Oman

[2] Department of Electrical and Electronics Engineering, Global College of Engineering and Technology, Muscat, Sultanate of Oman

[3] Department of Computer Engineering, University of Technology and Applied Sciences, Muscat, Sultanate of Oman

[4] Department of Mechanical Engineering, Faculty of Engineering, Universidad Tecnológica Metropolitana, Av. José Pedro Alessandri 1242, Santiago 7810003, Chile

1 Introduction

Cloud storage is becoming more and more popular because it provides a flexible on-demand data outsourcing service with alluring advantages, such as the elimination of the need for a storage management agreement, ubiquitous data access from any location, avoidance of capital expenditures on hardware, software, and individual maintenance, etc. However, this novel approach to data hosting also introduces fresh security risks to user data, leaving people and businesses with lingering uncertainties [1, 2]. Data availability is rising along with the number of cloud users. As a result, a large amount of space is needed to store these data. On a big scale, data compression approaches are mainly selected to lower the data size with the least amount of disturbance. The potential response to this problem is deduplication [3].

Consequently, storage space difficulties are handled through the data deduplication technique, which can improve the efficiency of storage space. The majority of the time, chunks and files are represented by hash values (fingerprints), which may be compared to other hash values to determine whether or not the chunk or file they represent is a duplicate [4]. Data duplication simply refers to compressed repeated data [5]. It is widely used for cloud storage servers, reducing the amount of unused RAM and maximizing bandwidth. It is advised to employ the Efficient Hash Function-based duplication detection technique before outsourcing the content in a cloud environment [6].

However, deduplication and data encryption are incompatible. Since images must be recognized based on their visual content, existing deduplication techniques for normal files cannot be used. Using a new perceptual hash algorithm called on Local Binary Pattern, [7] presents a safe imagery deduplication approach that combines hashing and clustering. A deduplication system that is effective and uses a weighted method is suggested [8] to conserve cloud storage space. Information security, information integrity, and unauthorized user access have all been problems with cloud data storage. Data redundancy and security problems come from the highly scalable and economical distribution and storage of information among many users. To solve these issues, service providers develop the bio-key for identification utilising a cryptographic technique, and only authenticated users will have access to it [9, 10].

The handling of patient health records (PHRs) data efficiently and securely is difficult for the healthcare sector. Fu et al. [11] have suggested a secure and reliable healthcare data service with application-aware deduplication on cloud-cover encrypted storage. In [12], the modular distributed file systems (DFSs) design, which makes it easier to use data reduction schemes with current programming techniques, presented an as-yet-unexplored option. Data compression methods are necessary for big-data processing systems like Hadoop, which often use DFSs, to enhance storage space efficiency.

As the demand for data sharing and intricate access-control policies continues to rise, traditional encryption systems, which are frequently built utilising public key infrastructure, struggle with high processing overheads and enormous network bandwidth usage. In the past few years, Attribute-Based Encryption (ABE) methods have garnered a lot of interest as a potential solution to these problems [13]. The cost of storage is decreased by minimizing duplicate content in cloud storage. Since the cloud server is often seen as being unreliable since a third party manages it, all data saved there is always encrypted before being uploaded, which has an impact on de-duplication due to the randomness property of encryption. Even when ownership is susceptible to dynamic ownership changes, the cloud server is permitted to restrict access to sourced information [14].

The motivation behind this research stems from the increasingly critical role of secure and efficient data management within the realm of cloud storage. As organizations and individuals alike rely more heavily on cloud service providers for data storage and transmission, the need to ensure both privacy and optimal resource utilization has become paramount. Traditional methods of data deduplication, while effective, encounter challenges when coupled with encryption requirements for enhanced security. This conflict between data encryption and deduplication efficiency prompted the development of a novel approach, which forms the basis of the proposed model. The core concept of this model involves the integration of a hybrid chunking algorithm, specifically the Content-based Two Threshold Two Divisor (TTTD) in conjunction with the Dynamic Prime Coding (DPC) technique. This combined approach aims to strike a balance between security and deduplication efficiency by adaptively handling different security levels for distinct data chunks. Moreover, the incorporation of an indexing method into the deduplication matching process seeks to enhance throughput and resource utilization, further contributing to the model's effectiveness. As an additional layer of security, a cross-domain authentication technique is introduced to safeguard data integrity and user privacy in open or semi-trusted systems like public clouds. This comprehensive model not only addresses the existing challenges but also introduces novel techniques that collectively advance the state of the art in cloud storage data deduplication.

The article is structured as follows. The literature review is covered in Sect. 2, while the problem statement and motivation are covered in Sect. 3. A research technique is discussed in Sect. 4, and experimental work is continued in Sect. 5. The conclusion marks the end of Sect. 6.

2 Literature survey

In the realm of data deduplication and security, several notable approaches have been introduced, each striving to address specific challenges. Naveen Kumar et al. [15] introduced the MTHDedup deduplication algorithm based on the Merkle hash tree, effectively enhancing data security by thwarting internal and external brute-force attacks. This method, however, does not delve deeply into the integration of encryption techniques to complement deduplication efforts. A similar concern arises in the approach proposed by [16], which utilized the Improvised Residue Number System and elliptical curve cryptography for encryption, aiming to prevent harmful uploads and downloads. However, the method's performance under dynamic ownership changes remains unexplored. Yao et al. [17] proposed a novel routing method based on Feature-Aware Stateful, effectively reducing system overhead while maintaining high deduplication ratios in distributed settings. This approach, although promising in optimizing system resources, lacks a comprehensive analysis of its security implications. Verma et al.'s [18] Dual-TEE approach focused on client-side protected deduplication within Trusted Execution Environments (TEE), offering security enhancements while optimizing storage and bandwidth utilization. However, the effectiveness of this method in scenarios with substantial dynamic ownership shifts requires further investigation.

While convergent encryption before outsourcing was suggested, its limitations in scenarios with dynamic ownership changes were evident. Ma et al. [19] server-side deduplication technique within a hybrid cloud environment demonstrated dynamic ownership control and resilience against collusion and copycat forging attempts. Nevertheless, the scalability of this approach and its potential under a broader range of cloud architectures could be explored in more depth. A method for duplicate data removal from clouds was introduced [20], highlighting increased security and resource efficiency. The challenge lies in addressing dynamic ownership scenarios and the effectiveness of the approach in various data-intensive cloud environments. Gnana Jeslin and Mohan Kumar [21] proposed a decentralized symmetry deduplication architecture, efficiently ensuring data ownership and accuracy in a cloud environment. However, the method's scalability and applicability to different cloud deployment models require further scrutiny. Athira et al. [22] aimed to overcome upload bandwidth, data storage, and integration challenges with their secure data storage system in DCS based on data portability and deduplication. Nonetheless, its performance under dynamic ownership and its suitability for diverse cloud ecosystems could be explored more comprehensively. The paradigm of Deduplication-before-encryption (DbE) introduced by Yang et al. [23] presented a unique angle, effectively reducing performance and storage costs associated with redundant data management. However, the trade-offs between deduplication and encryption need further evaluation to gauge the approach's robustness under various scenarios.

Rishabh Gupta et al. work [24] focuses on predicting malicious user behaviour using quantum properties, leveraging a developed malicious user predictor unit. This approach capitalizes on the computational and behavioural traits of Qubits and Quantum gates to accurately predict malicious user activities, enhancing the security of the system by up to 33.28%. Another contribution by Rishabh Gupta [25] introduces a robust data protection model in the cloud environment through data partitioning, partial decryption, and thorough analysis. This model not only fortifies data privacy but also remarkably improves accuracy, precision, recall, and F1-score, achieving relative improvements of up to 45.58%, 49.31%, 45.58%, and 48.46% over existing methodologies. Similarly, Rishabh Gupta et al.'s novel secure data protection model [26] offers enhanced privacy preservation through effective data partitioning, analysis, and protection, achieving significantly improved performance metrics compared to prior approaches. Gupta et al. [27] presented an innovative model for secure data sharing among multiple owners in a cloud environment, emphasizing differential privacy and machine learning techniques. This model enhances privacy preservation and performance, exhibiting high accuracy, precision, recall, and F1-score, with improvements of up to 23.33%. Additionally, Ashutosh Kumar Singh et al. [28] proposed a pioneering method for secure data sharing, employing data partitioning and noise injection techniques. Their approach showcases commendable accuracy and protection metrics, offering relative improvements of up to 29%. In a distinct realm, Muthunagai et al. [29] system design explores efficient time series data computation and deduplication in the cloud using Merkle Hash Tree and the Modified Distribution method. This system contributes to space and computation overhead reduction while augmenting data retrieval efficacy. Despite their advancements, these studies can further benefit from explicit discussions contrasting their models with existing works and addressing potential scalability challenges. Table 1 shows the key features of the review literature.

Table 1 Key features of the review literature

References	Objective	Key features	Outcomes	Limitations
Gang et al. [15]	Enhancing security in hybrid cloud deduplication	Merkle hash trees, unpredictable cipher texts, resistance to brute-force attacks	Improved data security against internal and external attacks, reduced computational burden	Limited scope in the hybrid cloud environment, potential scalability challenges
Lenin et al. [16]	Secured storage system with deduplication and cryptographic approach	ECC encryption, residue number system deduplication, cosine similarity checking	Prevents malicious uploads, effective data backup security	Limited discussion on scalability and resource efficiency
Yao et al. [17]	Feature-aware stateful routing for distributed deduplication	Feature-based node selection, stateful routing algorithm, local deduplication	Reduced system overhead, improved deduplication ratio, good load balance	Need for further evaluation with larger datasets and diverse workloads
Verma et al. [18]	Client-side dual-TEE-based deduplication in the cloud	Dual TEE model, privilege sets, Proof of Ownership (PoW)	Enhanced data deduplication security, better model resistance	Limited insight into performance under varied real-world conditions
Mounika et al.	Convergent encryption-based authorized deduplication	Convergent encryption, differential user privileges, authorized duplicate check	Secure authorized deduplication, minimal overhead, insider and outsider attack resistance	Potential challenges in implementation at scale
Ma et al. [19]	Server-side deduplication with ownership management in hybrid cloud	Hybrid cloud architecture, dynamic ownership, collusion attack resistance	Improved security and deduplication, efficient resistance against attacks	Limited discussion on potential performance bottlenecks
Vignesh et al. [20]	Novel technique for duplicate data removal and cloud data security	Encryption key consistency, chunk-based encryption, user-specific access	Enhanced data security, reduced processing time	Limited evaluation under heavy workloads and scalability
Gnana Jeslin et al. [21]	Decentralized symmetry deduplication architecture	Decentralized architecture, hand printing-based network model, probabilistic keys	Improved global deduplication, enhanced data ownership management	Limited exploration of potential vulnerabilities
Athira et al. [22]	Secure data storage with deduplication and portability	Feature reduction, hashing algorithms, LF-WDO technique	Improved computation time, better response time	Limited validation with diverse datasets and scalability
Yang et al. [23]	Deduplication-before-encryption (DbE) with shielded storage	DbE with Intel SGX protection, frequency-based deduplication	Improved deduplication efficiency, security via Intel SGX	Limited exploration of potential performance overhead
Gupta et al. [24]	Malicious user prediction using Quantum properties	Quantum-based malicious user predictor, accurate prediction	Enhanced security through quantum-based prediction	Limited discussion on real-world scalability and performance
Gupta [25]	Secure data protection with partitioning and analysis	Data partitioning, privacy preservation, efficient analysis	High data privacy with improved accuracy	Limited scalability discussion and real-world testing
Gupta et al. [26]	Privacy-preserving model with data partitioning	Data partitioning, noise injection, k-anonymization	Improved data privacy with accuracy, precision, and recall	Limited scalability discussion and diverse dataset evaluation
Gupta et al. [27]	Differential privacy and machine learning for multi-owner data sharing	Multi-owner data sharing, classifier privacy, robust mechanism	Enhanced privacy preservation, high accuracy	Limited discussion on potential vulnerabilities
Singh et al. [28]	Data partitioning with noise injection for cloud data	Data partitioning, noise injection, classifier usage	Improved data protection with higher accuracy	Limited scalability discussion and diverse dataset evaluation
Muthunagai et al. [29]	Computation of time series data during index based de-duplication	Time series data computation, Merkle hash tree, modified distribution method	Reduced computation overhead, increased data retrieval efficiency	Limited exploration of potential scalability challenges

3 Research problem definition and motivation

The information de-duplication method is utilized to store a single example of excess information and dispenses with the copy information in the data centre. It is utilized to diminish the data centre's size and decrease the replication of information that was copied on the cloud. Information deduplication has turned into ware in enormous-scale stockpiling frameworks, particularly in information reinforcement and documented frameworks. In any case, because of the evacuation of excess information, information deduplication de-linearizes information situations and powers the information pieces of similar information articles to be isolated into numerous different units. In the starter assessment, found that de-linearization of the information situation bargains the spatial zone of information, with information readability, reduced throughput, and in some cases reduced productivity, which implements emergency level reduction and makes some mitigation frameworks less acceptable.

Deduplication innovation can be grouped into customer-side deduplication and server-side deduplication innovation. Since the customer-side deduplication innovation has a bit of leeway from the perspective of proficient utilization of data transmission, numerous examinations on the customer-side deduplication are being done. Notwithstanding the numerous points of interest of the deduplication innovation, deduplication innovation on basic information has caused some new security issues. Asset-based encryption (ABE) is used widely in dispersed management, where an information supplier can retrieve its encoded information to Cloud Ace affiliate and pass it on to users with express accreditations (or attributes). But the standard ABE architecture is not in favour of secure reduction, which forces it to discard duplicate copies of obscure data to save extra room and framework communication. This quality-based framework is introduced with a secure reduction in a crossbreed cloud setting, where Open Cloud deals with responsibility and cut-off private cloud copy disclosure.

4 Proposed research methodology

By removing data copies from the cloud service provider (CSP), the approach based on data deduplication is employed in the storage cloud to reduce communication traffic and storage space. Data deduplication with secure data storage, however, is the key problem with cloud storage. In this massively data-driven society, secure information deduplication has the potential to significantly reduce the capacity overheads and correspondence in delivered storage benefits. The successful and secure deduplication of large amounts of information in distributed storage is examined in this article. Further, portray the consistency and trustworthiness model. With regards to malware examination, the learning model is set up in dataset existing verified malware models, naming risky or sympathetic level through the excellence of parallel classification, or sorting or aggregation of malware for multi-class classification. Figure 1 depicts the flow diagram of the proposed work.

The study suggests a hybrid technique that, while offering comparable deduplication performance, boosts chunking throughput. This eliminates boundary-shifting and operates based on the local highest value and a dynamic variable window. This research study proposes an arrangement to deduplicate mixed data set away in the cloud in light of ownership test and delegate re-encryption. In light of the TTTD calculation, it was therefore suggested to introduce a Content-Based Two Threshold Two Divisor (CB-TTTD) with a Multi-Level Hashing Technique to improve the deduplication process by accelerating the deduplication activity and increasing its pressure proportion. The results demonstrate inevitable efficiency and sufficiency in game plan potential balancing strategy, especially for enormous information reduction in spread accumulation.

4.1 Secure de-duplication process

The transferred documents belonging to the information owner may be stored in the cloud. Information deduplication is a specific information pressure technique utilised to eliminate redundant information that must be copied which is being produced. Before uploading a document to the

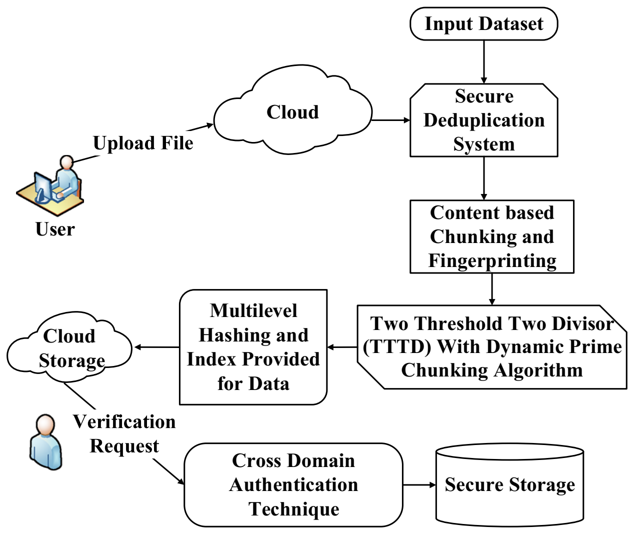

Input Dataset
Cloud
Secure Deduplication System
Upload File
User
Content based Chunking and Fingerprinting
Multilevel Hashing and Index Provided for Data
Cloud Storage
Two Threshold Two Divisor (TTTD) With Dynamic Prime Chunking Algorithm
Verification Request
Cross Domain Authentication Technique
Secure Storage

Fig. 1 Proposed work's architectural design

cloud, the proposed method requires the information owner to perform record-level and content-level de-duplication identification and counteraction. Deduplication of data is a technique for resolving storage issues. Hashing and indexing, Data chunking and matching stage are the four essential processes in the process of deleting duplicate data.

4.2 Chunking algorithm

Chunking, the initial stage of data deduplication, divides the data stream input (file) into manageable, non-overlapping units called chunks. For two strings with identical contents to give the same hash result, the chunking operation utilises a specific sort of rolling hash that is dependent on the text's actual contents. The chunks generated from this stage determine the deduplication ratio, making this stage extremely crucial.

4.2.1 Content-based chunking algorithm

The data deduplication chunking mechanism is content-defined chunking (CDC) based on the data's content chunking breakpoint is determined when a predetermined breaking circumstance is met. This is employed to solve the boundary-shifting issue that fixed-size chunking (static chunking) faces. To overcome the boundary-shifting problem, the suggested approach utilises a signal hash function for the CDC-based variable size that restricts the impact of adding or removing characters in a specific region.

4.3 TTTD with rolling hash

With a predetermined window size (48 bytes), the variable size-chunking method TTTD utilises the Rabin Fingerprint to determine the substring's hash value. The *mainD* performs the identical function as *D* did in the BSW algorithm. The *secondD*'s value is half that of the *mainD*. If the algorithm is unable to locate any breakpoints using the *mainD*, the *secondD* identifies a backup breakpoint due to its greater probability.

$$Rabin(B_1, B_2, \cdots B_a) = \left\{ \sum_{i=0}^{\alpha} \left(B_i * P^{\alpha-i-1}\right) \right\} Mod\, D \tag{1}$$

$$\left\{ \left[Rabin(B_i, B_{i+1}, \cdots B_{i+\alpha-1}) - B_i * P^{\alpha-1}\right] * P + B_{i+\alpha} \right\} Mod\, D \tag{2}$$

where *P* is a prime number, for the characters in the substring *Bx* is the ASCII code, *D* is the average chunk size and is the sliding window's size. For the first substring, the Rabin Fingerprint is calculated utilising Eq. (1). The first *W* bytes are sent to the hash algorithm, after which a sliding window removes the *W* size's first byte and adds the new one, updating the hash value or fingerprint. The fingerprint is considered the chunk boundary or breakpoint.

To calculate each tested data stream's fingerprints, CB-TTTD provides rolling hash functions. A rolling fingerprint is calculated over the data. The fingerprint is calculated utilising the following method for the first substring of length *K* where (*K* is Substring size):

$$\begin{aligned} &Fingerprint(Substring\, of\, length\, k) \\ &\quad = \left(Substring[0] \times Prime^{K-1}\right) + \left(Substring[0] \times Prime^{K-2}\right) \\ &\qquad + \left(Substring[0] \times Prime^{K-3}\right) \cdots \left(Substring[0] \times Prime^{0}\right) \end{aligned} \tag{3}$$

The fingerprint value is determined by applying formula (4) for the first string's each character with window size (36 bytes).

$$Fingerprint(B_0, B_2, \cdots B_a) = \left\{ \sum_{i=0}^{\alpha-1} Val[B_i] * 2^{i+1} \right\} \tag{4}$$

Applying (4), the fingerprint value is determined for each of the enumerated substrings.

$$\begin{aligned} &NewFingerprint(B_{i+1}, \cdots B_{i+\alpha+1}) \\ &\quad = [\{Fingerprint(B_i, \cdots B_{i+\alpha}) - Val[B_i]\} \div 2] \\ &\qquad + Val[B_{i+\alpha+1}] * 2^{\alpha-1}, \end{aligned} \tag{5}$$

where α is Substring Size and $B_1 \cdots B_\alpha$ are the substring characters, The value of index $[Bi]$ in the Fingerprint array is denoted by $Val[Bi]$. The character's value is selected from a 256-position array that corresponds to the printable characters and is filled with random values of (1, 2) to generate the array. To solve this issue, the system adds Dynamic Prime Chunking (OPC) to CDC, which utilises prime numbers to split up chunks and requires a minimal number of comparisons, fewer conditions, and less computational overhead.

4.3.1 DPC algorithm

The DPC algorithm's main objective is to dynamically split the input string into chunks based on prime integers. In the first step of the method, five predefined values, prime chunk value, prime chunk size, exponential value (e), average chunk size and chunk counter are predefined. Holding them ideally at the chunk boundary, within each piece, produces the highest value.

The third stage establishes the chunk boundary and maximum byte value according to the following two standards:

- Whether the maximum threshold value of *M* is more significant than all of the byte values inside the interval $[I, N]$ or the interval $[I, N]$ is likewise empty.

- The extreme value M in the dynamic variable-sized window cannot be less than the sum of all the bytes between $[O, C]$.

The algorithm then returns the step-four breakpoint position C after the chunk bounds have been declared. The implementation of the DPC method based on workflow is then covered in the paper. Table 2 shows the dynamic prime coding algorithm.

$A1$ stands for the first-byte position, and from there, it advances correctly until it reaches the byte position's end B. Performance won't be lost if the change is made before the extreme value. Accordingly, depending on chunk variance and bytes saved per second, the suggested approach enhances deduplication performance.

4.3.2 Multilevel hashing technique

As opposed to using the character's ASCII code, the multi-level hashing technique employs a value taken from the fingerprint array, which speeds up processing and lowers the CPU overhead required for each fingerprint (Table 3). The system tests several fingerprint array values, including

$[0,1], [1,0], [0,0,1,1], [1,1,0,0], [1,2],$
$[1,2,3,1,2,3], [1,2,3,4,\cdots 255]$

The sequence of [1, 2] values was the most effective value that generates a high deduplication ratio.

Table 2 Dynamic prime coding algorithm

```
Algorithm 1: DPC
Input: The input data string, Str; the file's remaining length, RL;
Output: Breakpoint for chunks, CPi
Prime_chunk_size [ ] = 2, 3, 5, 7, and 11 are predefined numbers.
        PRCH 5, // 5 is the prime chunk value.
          Chunk Counter is 0
          exponential value e = 2.718281828
          Average chunk size: 6 KB
DPC_CHUNKING (Str, RL) function
Calculate the size of a dynamic window.
DW is equal to 1024 * Prime_chunk_size [CC% PRCH] / (e-1)
CPi ← 1
Max_Value ← Str [CPi ].POS
Max_Position ← CPi
CPi ← CPi + 1
   While CPi < RL do
      if Str[CPi].POS < = Chunk_Avg_Size then
         if CPi = Max_Position + DW then
            return CPi
         end if
      else
         Max_Value ← Str[CPi ].POS
         Max_Position ← CPi
      end if
      CPi ← CPi + 1
   end while
return RL
end Function
```

Additionally, the data stream's characters are given weight via the CB-TTTD-Multi-Level Hashing Technique. In addition to the primary and secondary TTTD criteria, the algorithm also considers the Dot character ("."). A paragraph is considered to be a separate chunk when it begins with a "." character and ends with a space or a ";". Because computing the chunk boundary does not need any more processing steps, adding this condition boosted the deduplication ratio while maintaining the same chunking time.

The multilayer Hashing Technique proposes a novel approach to the collision problem that employs four hashing functions as opposed to one. The method will calculate and record the following four hash values for each chunk.

$$Hash\,1(Chunk) = \left\{\sum_{i=0}^{s-1}\left(Array1[Bi] * 2^k\right)\right\} \tag{6}$$

$$Hash\,2(Chunk) = \left\{\sum_{i=0}^{s-1}(String[Bi] * A1)\&0xFFFFFFFF\right\} \tag{7}$$

$$Hash\,3(Chunk) = \left\{\sum_{i=0}^{s-1}\left(Array3[Bi] * 2^k\right)\right\} \tag{8}$$

$$Hash\,4(Chunk) = \left\{\sum_{i=0}^{s-1}(String[Bi] * A1)\&0xFFFFFFFF\right\} \tag{9}$$

From the Eqs. (1, 2, 3, 4) above, s determines the chunk size, Array1 and 2 are arrays of 255, K is an integer value equal to (i mod 8), $A1$ and $A2$ are valued at 5 and 11, respectively, and their values grow by 1 with each iteration. The code $0xFFFFFFFF$ is used to obtain a constrained range of values. The reconstruction of the files will take advantage of this log file.

4.3.3 Matching process using index table

In deduplication, planning propels the structure must identify and discard the duplicated protuberances when another record arrives and passes the two sneak pinnacle compositions. As a result, another system that requires less calibration can be included in the system, enhancing throughput by reducing CPU usage and saving execution time.

Table 4 demonstrates the way the portioning approach distributes chunks based on the index value. This arrangement was tried with Datasets 1, 2 and 3 the impact of the first hash point of confinement is reduced to zero constant hash work. The test result reveals a significantly

Table 3 Fingerprint array

Array [0]	Array [0]					Array [255]
1	2	1	2	1	…	2

Table 4 Chunk distribution according to partitioning method

Index	From	To
0	0	462
1	463	471
2	472	481
3	482	491
4	491	503
5	504	515
6	516	530

noticeable improvement in the time required for the matching process.

4.4 Cross-domain authentication technique

The Web services approach presents frightening and inescapable security challenges. A better chameleon hash is employed in the intra-domain to redact the user's state, extended the blacklist Merkle tree in the inter-domain to guard against unauthorised access to the services of different domains, and the anonymity mechanism is utilised throughout the cross-domain authentication process to increase security and track malicious users.

4.4.1 Chameleon hash signature algorithm

Chameleon hashing, a non-interactive signature algorithm, evolved into the chameleon hash signature. The prover first chooses $s \in_R Z_q^*$ as its secret key. The prover then determines the chameleon as $C = G^s$. The prover chooses the private key $a \in_R$ and generates the appropriate public key as $P = G^a$ to carry out the authentication. The prover then generates an auxiliary parameter called $m_{au} = Cfind(a, nonce, s) = s - a\lambda$, where $Cfind$ is a collision detection method, s is the authentication challenge, and $\lambda = H(P \oplus nonce)$. When the verifier is given the prover's authentication data, it computes a chameleon hash parameter as $CH(m_{au}, P, nonce) = G^{m_{au}}.P^{\lambda}$. The verifier can properly authenticate the prover by determining whether the chameleon hash parameter equals C.

The proposed chameleon hash algorithm and a graph-based signature scheme are thus able to provide a practical chameleon signature scheme. The characteristics of the chameleon hash function:

- Anyone can readily measure the hash function utilising the public key (Hash key) based information.
- The hash function is collision-resistant if someone without access to the private key (trapdoor key) can't find the collision of two hashes.
- The consumer who is aware of the trapdoor key information, however, can identify the collision of the supplied inputs.

The trapdoor key CK_R is used to look for the corresponding collisions, while the hash key Hk_R is used to generate the chameleon hash function. If only HK is referenced in this way $h(m, r) = h(m\prime, r\prime)$, it is a difficult task to seek hashes based on collision on a few messages m,m' and two randomly picked numbers $r, r\prime$. When $m, m\prime$, and $r\prime$ are stated so that $h(m, r) = h(m\prime, r\prime)$, it is incredibly simple to determine r. Using a message m and a random number r to calculate $CH_R(.,.)$ may satisfy some of the following properties:

Collision Resistance $CH_R(.,.)$ on a pair of input messages m and randomised numbers $r(m1, r1)(m2, r2)$ where $m1 \neq m2$ is performed. There is no algorithm for determining the hash collision.

$$CH_R(m1, r1) = CH_R(m2, r2) \tag{10}$$

Trapdoor Collisions Deals with secret key information that, when combined with extra message $m2$ information, can determine value $r2$ on the inputs of $m1, r1$.

$$CH_R(m1, r1) = CH_R(m2, r2) \tag{11}$$

Uniformity All messages m will have the same probability distribution if random values r are selected uniformly.

As a result, for the suggested signature method to work, a minimum acceptable set's maximal antichain and an efficiently computed collision-resistant mapping from prerequisites include the hash space and the maximum antichain. The hash algorithm's process flow diagram is shown in Fig. 2.

The proposed HBCS system utilises the following algorithms:

Key Generation 1^k: The DAGs $G_i(i = S, V)$ is chosen by the signer S and the verifier V. A homomorphic pseudorandom generator should be utilised for PG which is connected to the expansion vertices and compression vertices of $G_i(i = S, V)$ and let H be a homomorphic hash function.

Sign sk_s, M: Let $M \in \{0, 1\}*$ be the message that needs to be signed.

$$\lambda = Ext_{\{\mu(z\prime)\}}[sk_s : G_s(e_s)] \tag{12}$$

Verify (pk_s, M, λ): The verifier computes

$$\text{Verifies} pk_s Ext_{\{e\prime\}}[\lambda : G_s(\mu(z\prime))] \tag{13}$$

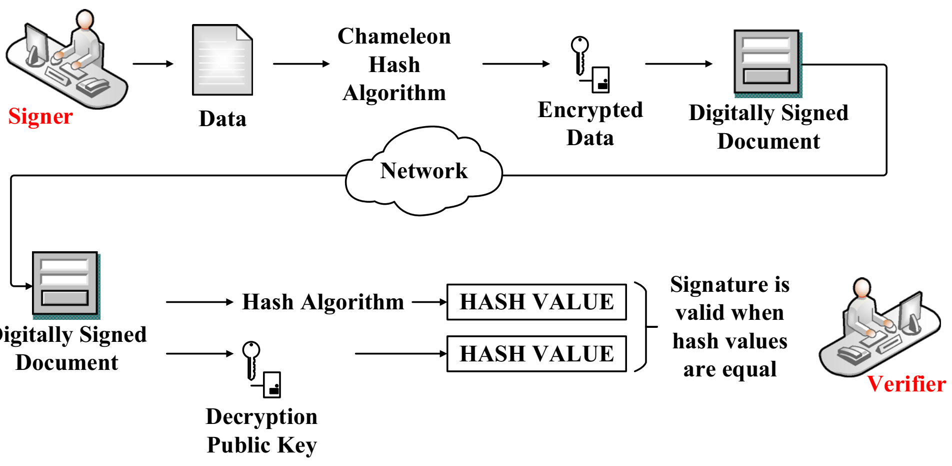


Fig. 2 Chameleon hash signature algorithm process flow

The suggested signature technique, therefore, requires the maximal antichain of least acceptable sets as well as a quickly computed mapping that resists collisions from the maximum antichain hash space. It also resists a variety of attacks.

4.4.2 Blacklist Merkle tree for verification process

To implement PKI-based signature solutions, more trusted parties are required, and there needs to be a reliable method of sharing certificate status information. Blacklists of revoked documents that bear signatures with a specific CA private key are known as Certificate Revocation Lists (CRLs). The client is ensured of the validity of the transition of the object's content within the time interval $[t_1, t_2]$ by checking the history of the write operations executed on the object. Finally, the clients themselves can converse with each other and find out whether any client's operation is missing from the action history to avoid the server from hiding the clients' write requests on the objects it stores.

Assume that there are m_1, m_2 branches in each of the two types of Merkle trees, which is satisfied by $m_1 = (m_2)^2$, to investigate the link between m and sum(m).

(i) *Verify Shards* The calculation timings of each shard are 1 $1(m) = log_{mn}$, and the latency of $verifyingshards = calculationtimes \times timecost$ of each calculation. $F1(m)$ reduces as m rises when n is fixed. As m increases, the latency of confirming shards also reduces.

(ii) *Generate Merkle Trees* The dimensions of the public tree directly affect how quickly Merkle trees are generated. Learn from the previous that when m rises, the public tree's overall node count decreases. As m increases, the latency of producing Merkle trees also reduces.

(iii) *Generate Auxiliary Path* The dimensions of the auxiliary path directly influences the speed at which it can be generated. The auxiliary path is $F2(m)$ in length.

A hash function transforms any kind of arbitrary data, regardless of length, to a fixed-size output. Given that it is a cryptographic function, it is frequently employed in cryptography. They are effective and well-known for having one characteristic: irreversibility. For instance, value $H(v_1||v_2)$ will be allocated to a node that has two children with the values v_1 and v_2, respectively. A sampling problem is thought to exist when only a data shards subset is chosen for validation. Repetition of sampling (sampling with replacement) to make sure that each shard is equally likely to be selected, which ensures fairness.

5 Experimentation and results discussion

This section offers a study of the performance statistics for the suggested content-based data deduplication techniques and TTTD-DPC. Following are the performance metrics: throughput, chunking and processing time, and total chunk count. Table 5 displays the dataset that was utilised to run the algorithm.

The information for Datasets 1 and 2 came from the programming plans of FIU's mail server and web server, respectively. Vdi images of five Linux virtual machines running in the lab from Dataset 3. The degree of reduction in the table typically lasts at a single focus point and is around the size of a 4 KB square. The DRs of Dataset 3 utilising Fixed-Size Isolating (FSP) and Content-Ported Lumping (CDC) are 2.06 and 2.36, respectively.

Table 6 illustrates the machine configuration. With 32 GB of memory and an Intel Gigabit 100 Mbps network system running Ubuntu, this is functional. SHA-1, Adler, and Rabin fingerprint are utilised, together with two

Table 5 Characteristics of dataset

Dataset	Number of files and folders	Total size
Versions of Emacs of GNU	16,296 files, 327 folders	580 MB
Versions of 3DLDF of GNU	5795 files, 63 folders	1.27 GB
Vdi pictures of five Linux virtual machines	–	

Table 6 Configuration of machines

Item	Configuration
CPU	Intel Xeon E5-2620@2.00 GHz
Memory	32 GB
Disk 1	TB 1*2
Network	Intel Gigabit 100 Mbps
Operating system	Ubuntu 12.04 64 bit
Kernel	Linux 3.5.0-23-generic
Operating environment	Jdk1.7

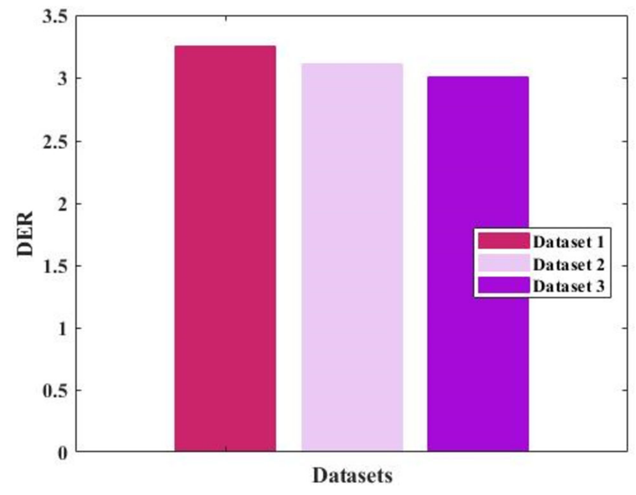


Fig. 3 Deduplication elimination ratio on different datasets

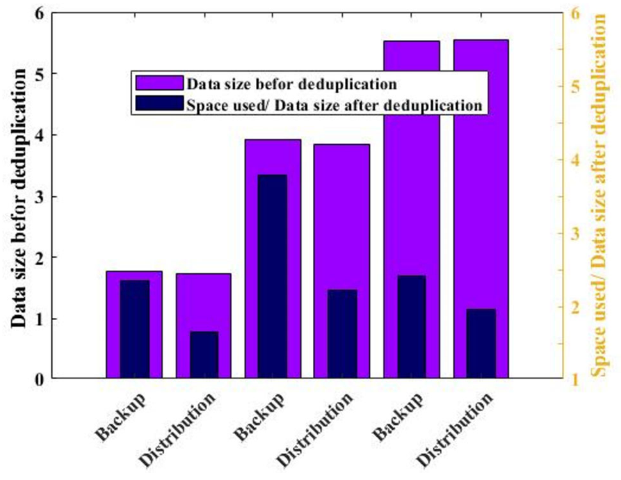


Fig. 4 Results of data size analysis after deduplication

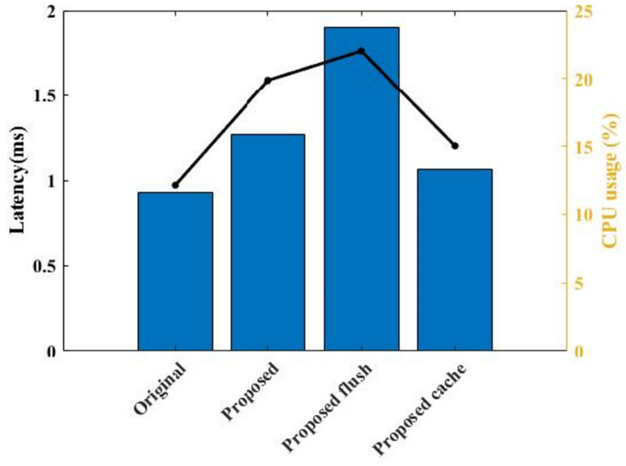


Fig. 5 Write time differences with proposed algorithm

separate datasets: To compare the effectiveness of the suggested system with other ones, versions of the GNU Emacs editor and the 3DLDF of GNUTTTD chunking algorithm with Rabin fingerprint are both implemented in the same context.

The article has examined the similarity degree and the duplicate-elimination measure of the similarity-only deduplication approach on three datasets, as seen in Fig. 3. The three datasets represent Versions of 3DLDF of GNU, Versions of Emacs of GNU and vdi pictures of five Linux virtual machines respectively. Dataset 1 has a higher elimination ratio of 3.3 when compared to the other two datasets.

The experimental data volume following duplication of the backup or primary storage system is shown in Fig. 4 together with the distributed storage system. The findings demonstrate that adjusting the dimension of the chunk among backup and storage for both before and after deduplication yields the best storage volume after removing duplication.

The variations in latency times among the suggested algorithms are seen in Fig. 5. Because some cache blocks in the disabled/enabled subarrays are flushed during dynamic resizing, the mass ratio increases. To do this, the cache's minimum write time for files 1.2 latency is for files. It appears to be using the most CPU compared to the original file, at around 15%.

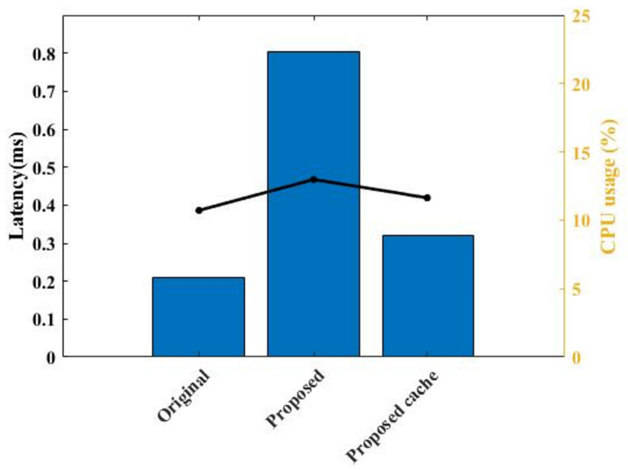


Fig. 6 Read time differences with proposed algorithm

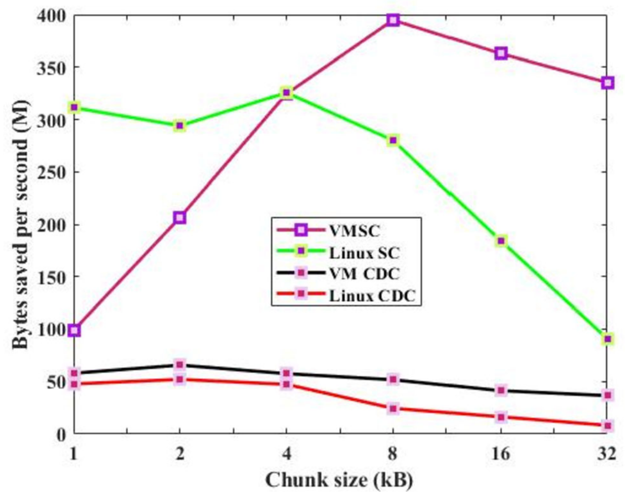


Fig. 7 Chunk size effects on deduplication effectiveness

The proposed cache's read time compared to Fig. 6 displays the original file for clarity. Performance is directly impacted by latency errors, as static resizing frequently misses the mark and is not ideal. With 8% CPU usage in this case, the suggested cache achieves a miss latency of 0.41 ms. Similarly, the proposed approach has a latency rate of 0.7 ms which is higher than the other two file documents with cache size.

Figure 7 utilises the highly redundant Linux SC, VM CDC, Linux CDC, and VMSC application datasets to test the effectiveness of a local–global source deduplication-based cloud backup service. The CDC scheme with an average chunk size of 2 KB can produce the best efficiency for the Linux dataset, the SC method with a chunk size of 8 KB can produce the best efficiency for the VM dataset, and the CDC scheme with an average chunk size of 4 KB can produce the best efficiency for the DOC dataset.

5.1 Performance metrics

Change the data/file size, the number of stored files, the deduplication rate, etc. to evaluate the overhead in the proposed deduplication system. Processing time, read time, chunking time, and write time are the four metrics that take into account the time difference. Additionally, the following section will discuss throughput and deduplication ratios individually as relevant metrics.

5.1.1 Chunk variance

Total chunk count Each file defines the total of the chunk counts in a backup path as the full chunk. It is represented by the following equation.

$$T_C = \sum_{F_{i=0}}^{TF} NCF_i \, TF \geq F_i, \tag{14}$$

where T_C defines the aggregate chunk count, NCF_i states how many pieces are in the file, TF denotes the number of files overall, and F_i states the starting of the file's index $F_i = \{1 \cdots TF\}$.

$$Average\,Chunk\,Size = \frac{Total\,Input\,Data\,Size}{Total\,Number\,of\,Chunks} \tag{15}$$

5.1.2 Time difference

The following section discusses the length of time to process backup data, including how long it takes to read, write, chunk, and process data.

Processing time Read, chunk, hash, dedupe, and write times are all included in the processing time.

Read and write time It describes the duration of a reading session and processes each file's backup data using various operations like chunk, hash, and dedupe. The collected unique chunks are then written to the persistent storage's container store.

Chunking time The amount of time, measured in seconds, needed to split the backup data into chunks of varying sizes. In this paper, CB-TTTD-DPC outperforms all other metrics in terms of quick chunk times.

5.1.3 Deduplication ratio

The percentage of storage space that has been saved by utilising this suggested deduplication system is known as the deduplication ratio. The efficiency of deduplication is increased by this value. The following equation defines DR as the changes in the total data size (D_{SZ}) and the total data stored size (DS_{SZ}), divided by the data size (D_{SZ}).

$$Deduplication\,Ratio\,(DR) = \frac{D_{SZ} - DS_{SZ}}{D_{SZ}} \tag{16}$$

5.1.4 Deduplication gain

It shows the amount of original content there is in the dataset. It is computed as in (13) in this study.

$$Deduplication\,Gain = \frac{Size\,of\,Deduplicated\,Data\,Detected}{Total\,output\,Data\,Size\,After\,Deduplication} \tag{17}$$

5.1.5 Throughput

The following equation calculates the throughput performance of OPC by dividing the total number of chunks processed in a file by the chunking time employed.

$$Throughput = \frac{T_c}{Chunking\,Time} \tag{18}$$

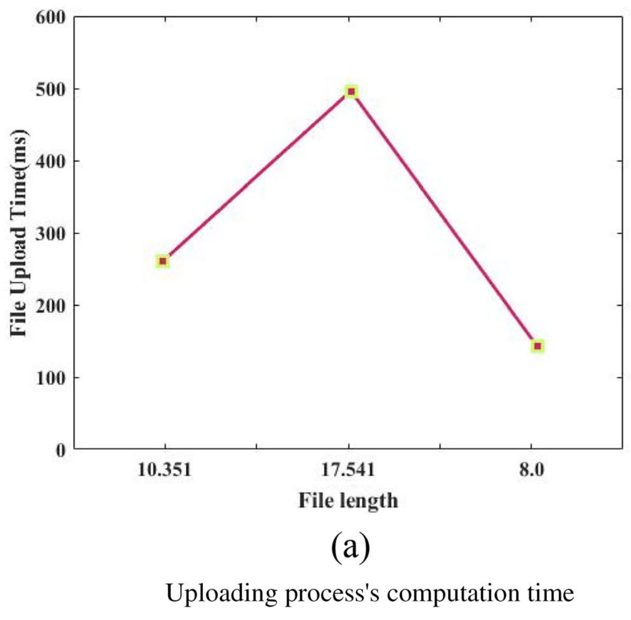
The accomplishment of the suggested technique for the aforementioned metrics on three datasets is shown in Figs. 6, 7, and 8. In terms of chunk variance, time variance DR, and throughput, it can be deduced from the results that the CBTTTD-DPC performs better than other CDC for all types of datasets.

5.2 CRP length's effect on the deduplication ratio

The tests are conducted from the following angles; the first one examines and analyses whether the suggested system behaves about the divisor length (i.e., the extent of the CB-TTTD-DPC fingerprint). Therefore, when trying to utilise the most related breakpoints among different files in the tested dataset, the proposed CB-TTTD-DPC clearly, fingerprint length matters in chunking technique fulfilment.

The computation of upload and download times dependent on file length is seen in Fig. 5. Figure 8a demonstrates the uploading process's computation time, and Fig. 8b displays the downloading process's computation time. In addition, the download's computation time procedure takes 22 s for document size 1 and 30 s for document size 2, which is 25,000 kb. With a file length of 8.0, downloading takes the least amount of time compared to uploading.

The case requires a higher conditional statement than AE, as demonstrated by TTTD. As a result, the computation overhead utilising CPU resources automatically increases. The table mostly displays our efforts. The purpose of OPC is to accelerate processing by focusing on tasks that require the fewest computer resources. As seen in Table 7 above, the OPC algorithm is framed without comparison.



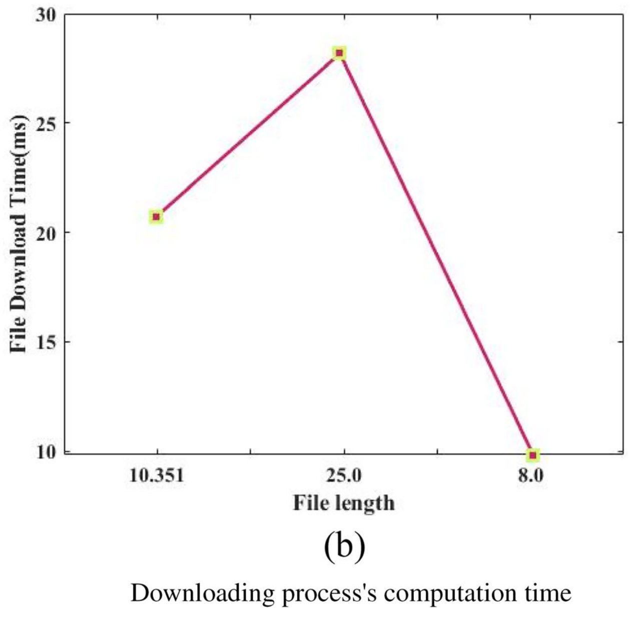


Fig. 8 Time analysis for file downloading and uploading

Table 7 Computational overhead and time difference comparisons for CDC

Algorithm	Computational overhead	Processing time
Rabin	1 OR, 2 XOR, 2 shifts, 2 array lookups, 1 conditional branch	4547
AE	1 comparison, 2 conditional branches	5012
TTTD	1 comparison, 5 conditional branches	5276
Proposed hybrid technique	2 conditional branches	3945

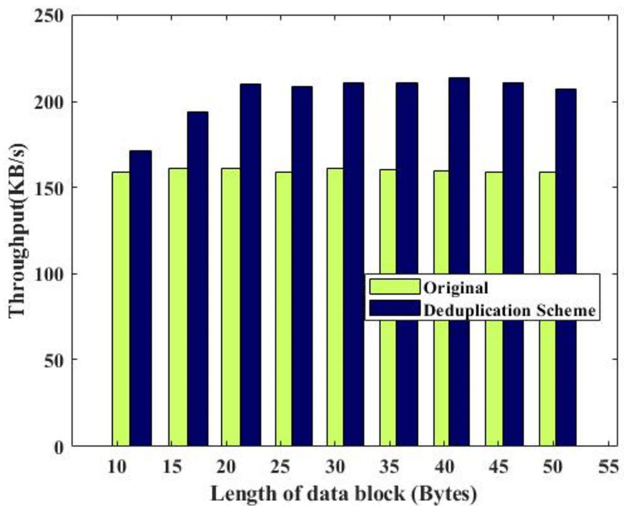


Fig. 9 Throughput vs length of data blocks

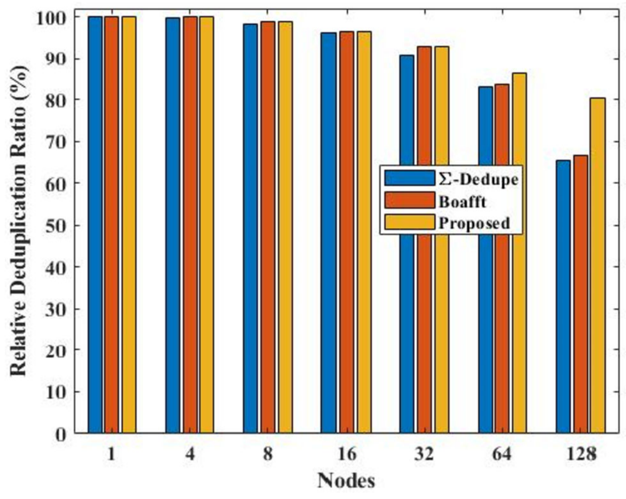


Fig. 11 Relative deduplication ration with different sizes

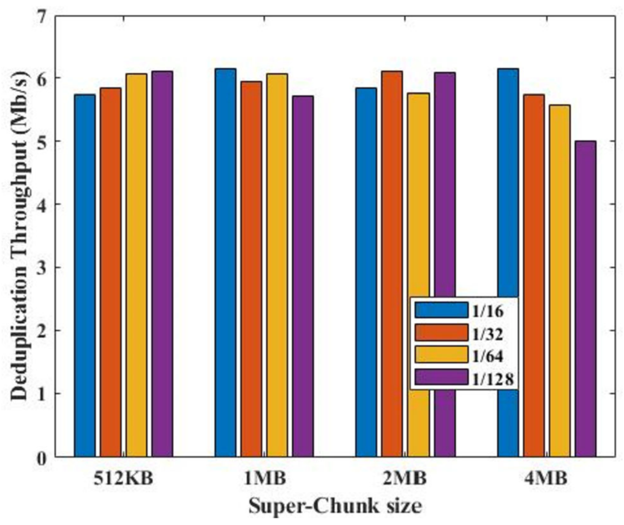


Fig. 10 Effect of superblock dimension and throughput for deduplication

5.2.1 Chunking throughput

By dividing the amount of data processed and the total time required to chunk the files, the chunking throughput has been calculated. The study evaluates the chunking algorithm and the individual datasets for a straightforward description. In comparison to alternative CDC algorithms, the study found that DPC offers faster throughput of 200 KB/s for Linux and compressed files.

The effect of data block length on throughput for the original and deduplication schemes is shown in Fig. 9. The highest throughput is achieved with blocks that are ten bytes in size. The deduplication strategy provides the highest throughput when compared to the real data block, with a range of 160–220 KB/s.

Figure 10 displays the throughput at various sampling rates and super-chunk sizes. The outcome demonstrates that the highest throughput of the system is achieved. When the super-chunk size is 1 MB and the sampling rate is 1/16. The handprint is 16 when the sample rate is 1/16. Because of this, the model adjusted the super-chunk size and sampling rate to 1 MB and 1/16, respectively, making the handprint size 16.

5.2.2 Relative deduplication ratio (RDR)

RDR is the comparison between the deduplication ratios achieved by global matching and similarity matching

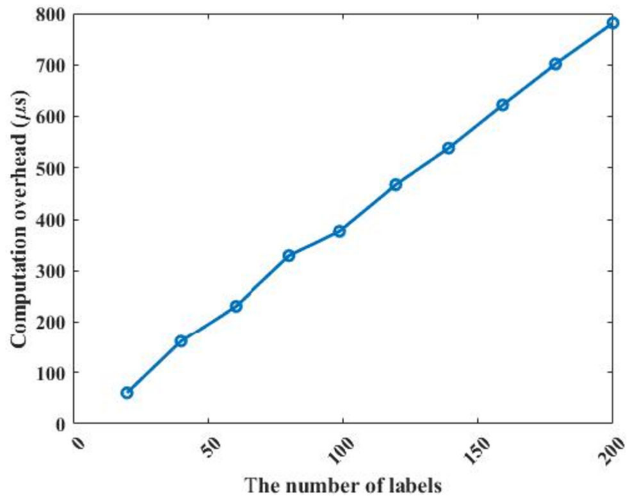


Fig. 12 Communication overhead vs number of labels

inside a node. The study utilises the abbreviations DR_g and DR_s to denote the global deduplication ratio and the similarity-matching-derived deduplication ratio, respectively. In light of this, the RDR can be expressed as follows:

$$RDR = \frac{DR_g}{DR_s} \tag{19}$$

The suggested work makes utilises the distribution and backup data features in nodes to quickly choose K candidate nodes and guarantee data similarity between the K nodes and the data. To prevent an imbalanced system load, this model takes node storage consumption into account.

Figure 11 shows that the deduplication ratio for $\sum$-Dedupe is lower than that for Boafft's and suggested. As illustrated in Fig. 5a, $\sum$-Dedupe and Boafft may achieve a relative deduplication ratio of roughly 90% in the Linux dataset when the cluster size is minimal. The maximum RDR of around 80% is feasible. Therefore, when the cluster size is large, the recommended technique can somewhat improve RDR.

The computational overhead rises as raising the number of labels, as depicted in Fig. 12. In particular, the time required to build the label tree is just 772 microseconds even when the amount of label nodes approaches 200 (near the number of labels needed by major organisations).

The deduplication ratio for various techniques is shown in Fig. 13. For the observation's conventional algorithms on datasets 1 and 2, a slight divergence in the ratio has been found. Comparison analysis demonstrates that the suggested solutions result in a lower deduplication ratio. As a result, content-based TTTD by default offers minimal DR and low chunk density.

The throughput capability for different datasets using CDC's four algorithms AE, TTTD, OPC and Rabin is shown in Fig. 14. The findings show that all algorithms

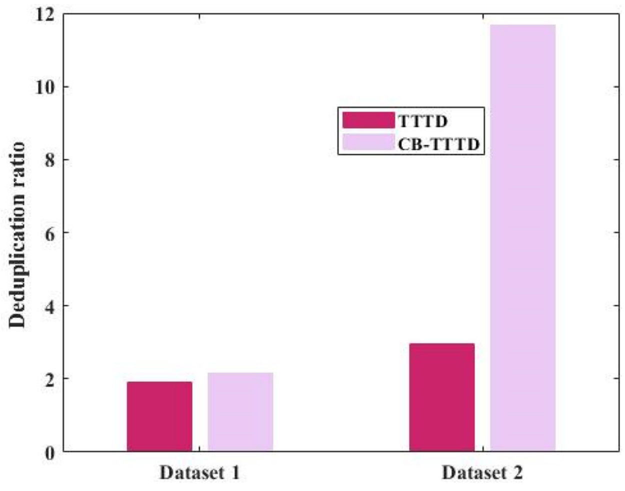


Fig. 13 Duplication ratio comparative analysis

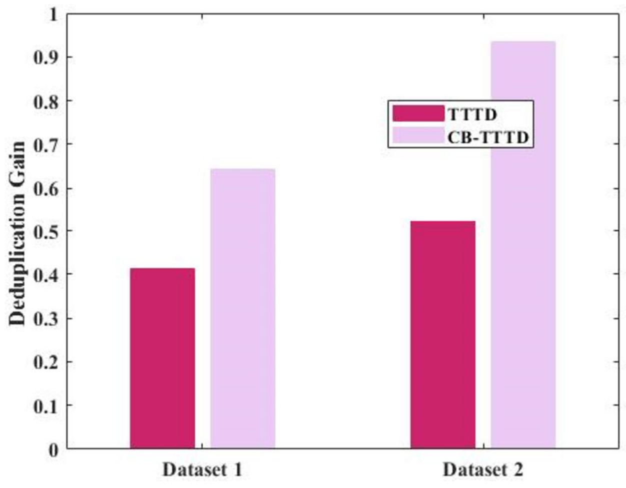


Fig. 14 Deduplication gain comparative analysis

have worse throughput for image datasets. The three dataset deduplication gain and ratio data values are shown in Table 8.

Figure 15 shows the chunk creation time on average of the proposed technique and the existing SHA-1 technique. Results indicate that the suggested mechanism has a faster average chunk formation time of 0.079–0.077 when compared to the existing techniques of SHA-1 and MD-5 in this figure, where the data size ranges from 400 to 1000 MB.

The effect of hash algorithms on the hashing stage time calculated by the aforementioned equations is shown in Fig. 16. As compared to the existing method, the proposed algorithm has a reduced hash creation time with data files of (400 MB, 600 MB, 800 MB, and 1000 MB). At 400 MB the average hash creation time is 0.074.

Figure 17 displays the throughput for deduplication obtained from the data stream under different values of (2 stream–128 streams), and further compares them with that

Table 8 Gain and ratio of deduplication for three different datasets

Dataset	Original size (GB)	Reduced size (GB)	Deduplication gain	DER
Dataset 1	13	4	3.25	0.692
Dataset 2	14	4.5	3.11	0.678
Dataset 3	15	5	3.00	0.666

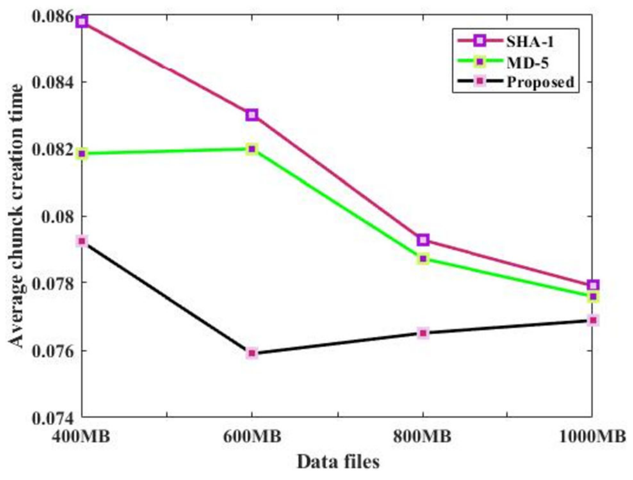


Fig. 15 Chunk creation time on average

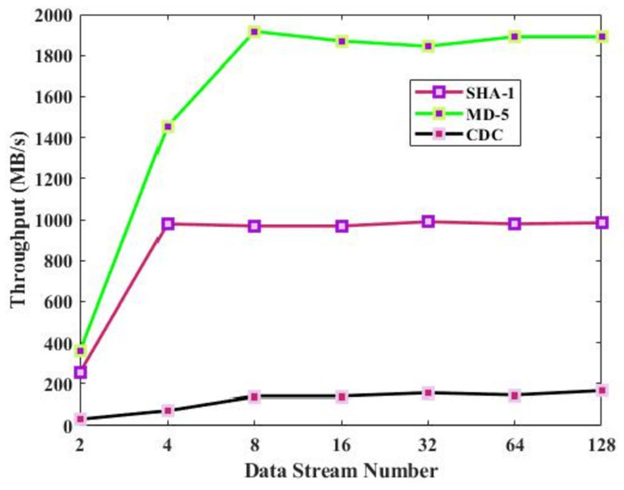


Fig. 17 Comparative throughput analysis

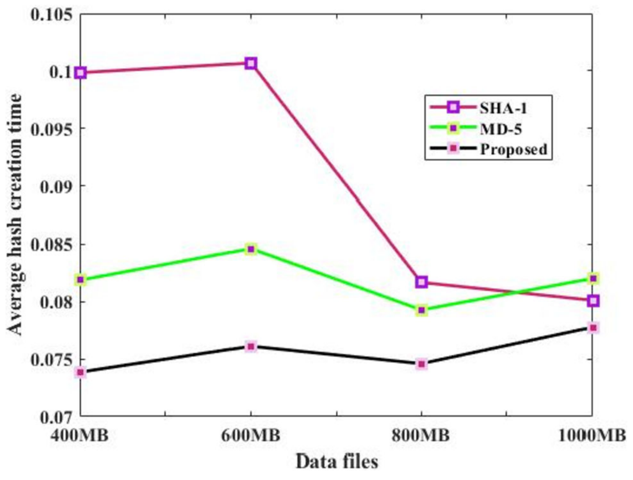


Fig. 16 Average hash creation time comparison analysis

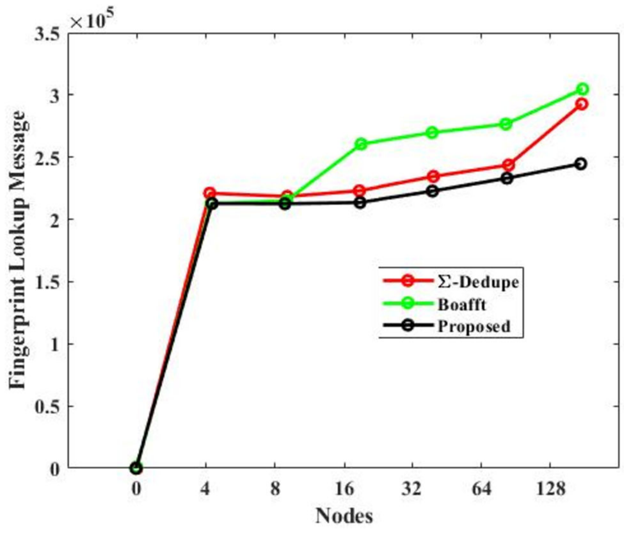


Fig. 18 Fingerprint lookup message with different nodes

of SHA-1, MD5 and CDC approach. The throughput of the proposed technique is obtained as 1900 Mb/s. The proposed deduplication throughput is determined to be less than that of existing approaches, as can be observed from the results.

The comparison analysis of fingerprint lookup messages with various nodes is shown in Fig. 18. As seen in Fig. 18, Boafft sends sample fingerprints of the superblock to each node to calculate similarity, hence its fingerprint search message is higher than that of suggested and $\sum$-Dedupe. Additionally, Boafft generates 1-to-all communication overhead by sending the fingerprint lookup requests to every node.

The comparison findings for storage efficiency are displayed in Fig. 19. The efficiency of storage is 400 and decreases as the amount of data blocks the cloud server holds increases. The efficiency in deduplication of the scheme is the highest among these three plans, and as a result, the storage effectiveness of the scheme achieves the highest performance.

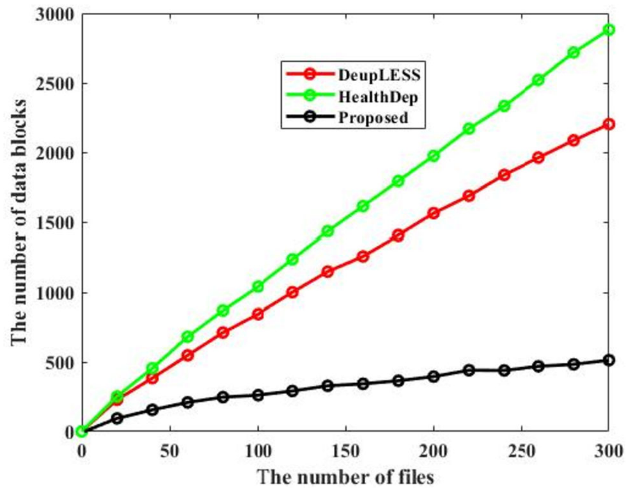


Fig. 19 Cost of storage in the data upload phase comparison

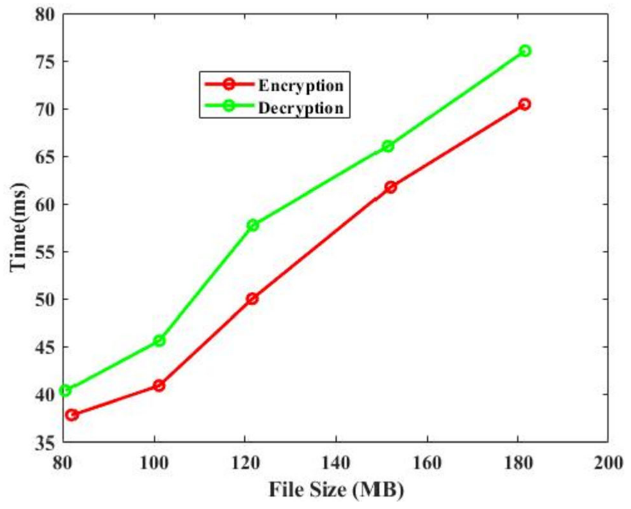


Fig. 20 Data encryption and decryption

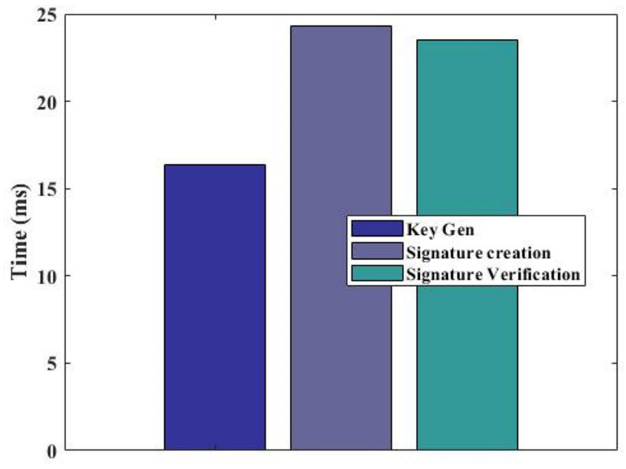


Fig. 21 Data ownership verification

Figure 20 illustrates the data encryption and decryption time. Information encryption and decryption were finished by utilizing the chameleon hash algorithm. The activity time was assessed utilizing CH key size 256 bits for various information sizes (extend from 80 to 150 MB). Information encryption and unscrambling were effectively done by utilizing the chameleon hash algorithm.

For each progression in the information proprietorship check, the activity time was evaluated utilizing 2048-piece BMT. The time taken for a key age was 16 ms as depicted in Fig. 21. The time taken for the production of the mark was 24 ms. The time taken for signature verification is 23 ms. In the suggested pattern, the information holder did not have to continue for information encryption if that information was at that point transferred by another information holder.

The comparison of the security of safe data deduplication algorithms is shown in Table 9. 'Satisfied', 'Not satisfied,' and 'Partially satisfied,' respectively, are indicated by SA, NS, and PS. In terms of the following qualities, the study contrasts the proposed plan with three other schemes [30–32]. Furthermore, the Yang system does not ensure protection against attacks carried out in cooperation with dishonest users. Shin's method successfully thwarts side-channel assaults on all properties [33, 34]. As a result of this method, it is evident that the suggested system is more secure than others [35].

6 Discussion

The article has examined the similarity degree and the duplicate-elimination measure of the similarity-only deduplication approach on three datasets, as seen in Fig. 3. The three datasets represent Versions of 3DLDF of GNU, Versions of Emacs of GNU, and vdi pictures of five Linux virtual machines respectively. Dataset 1 has a higher elimination ratio of 3.3 when compared to the other two datasets. This substantial difference in the elimination ratio could be attributed to the nature of the data within Dataset 1. Dataset 1 may contain a higher degree of redundancy or similarity among the files, leading to a more effective deduplication process. The experimental data volume following duplication of the backup or primary storage system is shown in Fig. 4 together with the distributed storage system. The findings demonstrate that adjusting the dimension of the chunk among backup and storage for both before and after deduplication yields the best storage volume after removing duplication. This result highlights the impact of optimizing the chunk size for deduplication. By tailoring the chunk size to the specific dataset, the storage system can achieve more efficient deduplication and reduce the overall storage footprint. The variations in latency

Table 9 Compared safe data deduplication techniques

References	Without KS	Confidentiality	Resistance to brute force attacks	Security against collusion	Side channel attacks
Li et al. [30]	NS	Edge: X, KS: PS	SA	NS	NS
Yang et al. [31]	NS	SA	SA	NS	NS
Shin et al. [32]	SA	SA	SA	SA	NS
Proposed	SA	SA	SA	SA	SA

KS key server, *NS* not satisfied, *SA* satisfied, *PS* partially satisfied

times among the suggested algorithms are seen in Fig. 5. Because some cache blocks in the disabled/enabled subarrays are flushed during dynamic resizing, the mass ratio increases. To do this, the cache's minimum write time for files 1.2 latency is for files. It appears to be using the most CPU compared to the original file, at around 15%. The observed increase in CPU usage during dynamic resizing of cache blocks could be attributed to the overhead of managing the cache state and adapting to changing cache sizes. This additional computational load during dynamic resizing can lead to the observed higher CPU utilization. The proposed cache's read time compared to Fig. 6 displays the original file for clarity. Performance is directly impacted by latency errors, as static resizing frequently misses the mark and is not ideal. With 8% CPU usage in this case, the suggested cache achieves a miss latency of 0.41 ms. Similarly, the proposed approach has a latency rate of 0.7 ms which is higher than the other two file documents with cache size. The difference in read time latency between the proposed cache and the other methods can be attributed to the dynamic resizing strategy employed. The higher latency rate in the proposed approach might be the result of the trade-off between adapting to changing cache sizes and maintaining lower latency. Figure 7 utilizes the highly redundant Linux SC, VM CDC, Linux CDC, and VMSC application datasets to test the effectiveness of a local–global source deduplication-based cloud backup service. The CDC scheme with an average chunk size of 2 KB can produce the best efficiency for the Linux dataset, the SC method with a chunk size of 8 KB can produce the best efficiency for the VM dataset, and the CDC scheme with an average chunk size of 4 KB can produce the best efficiency for the DOC dataset. These results reflect the impact of chunk size on deduplication efficiency. The optimal chunk size can vary depending on the dataset characteristics, such as the nature of redundancy, file sizes, and patterns. The observed variations in efficiency highlight the importance of choosing appropriate chunk sizes to achieve optimal deduplication performance for different datasets.

7 Research conclusion

One of the rapidly developing areas of technology is cloud computing, which enables the storing, access, and execution of data, initiatives, and other information-related services over the Internet [36]. The information must be stored securely and delivered to many users safely while utilising cloud information services. Information security, information integrity, and unauthorised user access have all been problems with cloud data storage. Data redundancy and security challenges arise as a result of the highly scalable and economical distribution and storage of data across several users [37]. For edge-assisted storage cloud systems, the study in this paper developed a security-aware and effective data deduplication strategy. It not only boosts deduplication's effectiveness but also lessens data leakage brought on by frequency analysis assaults. To gauge the security requirements of users, the article specifies the security level for chunks initially. In light of this, the Content-based TTTD with the DPC algorithm is presented together with a multilevel hashing method and index table matching. Redundant data are eliminated and lowering bandwidth and storage costs, this strategy is used. Because fewer new storage devices are required, it is advantageous for cloud services. The four currently used algorithms, TTTD, Rabin, AE and MAXP have their metrics from various datasets analysed. The suggested study TTTD-DPC is contrasted with AE while presenting the findings. The primary reason is that AE is a quick chunking approach and achieves faster throughput. Because it avoids retracing and discovers the maximum value and chunks boundary. Similar to HPC, DPC offers a lower chunk variance and higher throughput. The testing also revealed that the proposed technique takes up 20 MB less space for the index table than SHA1 and MD5, which each require 65 and 81 MB. In comparison to the traditional hashing technique, the suggested function assumes straightforward mathematical operations and requires a computation time of only roughly 15 ms. Since server-side deduplication and proof-of-ownership protect against side-channel attacks and brute force attacks, respectively, the suggested technique is susceptible to a variety of attacks.

Further research could focus on refining the security-level specification for chunks, incorporating advanced encryption techniques, and exploring cryptographic methods that enhance protection against a wider range of potential attacks, including advanced cryptographic attacks. Leveraging machine learning techniques for anomaly detection and pattern recognition within the deduplication process could enhance the system's ability to detect and thwart sophisticated security breaches and unauthorized access attempts. Future work could focus on optimizing the efficiency of the deduplication process, particularly in terms of computational overhead, memory usage, and network bandwidth consumption, ensuring scalability and performance across various cloud scenarios. Exploring collaborative security approaches that involve multiple cloud service providers and stakeholders could lead to a more robust and comprehensive security framework for cloud data storage.

Appendix

Variable	Description
α	Substring size
$B_i, B_{i+\alpha}$	Substring characters
P	Prime number
D	Average chunk size
W	Window size
DW	Dynamic window size
s	Chunk size
Array1, Array2	Arrays of size 255
K	Integer value equal to (i mod 8)
A1, A2	Constants with values 5 and 11, respectively
e	Exponential value
RL	Remaining length of a file
CPi	Breakpoint for chunks
M	Maximum threshold value
I, N, O, C	Byte position indices
Prime_chunk_size	Predefined prime chunk values
Hash functions	Various hash functions
PRCH	Prime chunk value
Chunk_Avg_Size	Average chunk size
G, λ	Parameters for chameleon hashing
m_{au}	Auxiliary parameter for chameleon hashing
$\mu(z\prime)$	Value associated with input z′
$[t_1, t_2]$	Time intervals
m_1, m_2	Branches in Merkle trees
n	Parameter in Merkle tree generation
$F1(m), F2(m)$	Functions related to latency
H	Hash function
v_1, v_2	Values for hash calculation
Bx	characters in the substring
$Val[Bi]$	$[Bi]$ in the fingerprint array
$Cfind$	collision detection
m, m′	collision on a few messages
$r, r\prime$	randomly picked numbers
$CH_R(.,.)$	Collision resistance

Author contributions All authors contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Naveenkumar, R., Muthusamy, K., Prasath, N.A., Krishnaraj, R.: Deduplication and security enhancement in cloud computing. Int. J. Electr. Eng. Technol. (IJEET) **10**(6), 54–60 (2019)
2. Widodo, R.N.S., Abe, H., Kato, K.: HDRF: hadoop data reduction framework for hadoop distributed file system. In Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems, pp. 122–129 (2020)
3. Verma, G.: Secure client-side deduplication scheme for the cloud with dual trusted execution environment. IETE J. Res. (2022). https://doi.org/10.1080/03772063.2021.2017360
4. Ruba, S., Kalpana, A.M.: An improved blockchain-based secure data deduplication using attribute-based role key generation with efficient cryptographic methods. (2021). https://doi.org/10.21203/rs.3.rs-596633/v1
5. Jinila, Y.B., Jose, J., Vimali, J.S., Ajitha, P., Gowri, S., Sivasangari, S.: A primitive solution to avoid data deduplication in cloud and overcome security threats. In: 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), pp. 113–117. IEEE (2021)
6. Periasamy, J.K., Latha, B.: Efficient hash function-based duplication detection algorithm for data deduplication deduction and reduction. Concurr. Comput.: Pract. Exp. **33**(3), e5213 (2021)
7. Chen, L., Xiang, F., Sun, Z.: Image deduplication based on hashing and clustering in cloud storage. KSII Trans. Internet Inf. Syst. **15**(4), 1448–1463 (2021)
8. Elangovan, P., Sumalatha, M.R.: Weight based deduplication for minimizing data replication in public cloud storage. J. Sci. Ind. Res. **80**(03), 260–269 (2021)
9. Venkatachalam, K., Prabu, P., Almutairi, A., Abouhawwash, M.: Secure biometric authentication with de-duplication on distributed cloud storage. PeerJ Comput. Sci. **7**, e569 (2021)

10. Liang, X., Yan, Z., Deng, R.H., Zheng, Q.: Investigating the adoption of hybrid encrypted cloud data deduplication with game theory. IEEE Trans. Parallel Distrib. Syst. **32**(3), 587–600 (2020)
11. Fu, Y., Xiao, N., Chen, T., Wang, J.: Fog-to-multicloud cooperative eHealth data management with application-aware secure deduplication. IEEE Trans. Dependable Secur. Comput. (2021). https://doi.org/10.1109/TDSC.2021.3086089
12. Widodo, R.N.S., Abe, H., Kato, K.: Hadoop data reduction framework: applying data reduction at the DFS layer. IEEE Access **9**, 152704–152717 (2021)
13. Li, J.S., Liu, I.H., Lee, C.Y., Li, C.F., Liu, C.G.: A novel data deduplication scheme for encrypted cloud databases. J. Internet Technol. **21**(4), 1115–1125 (2020)
14. Sivasankari, S., Lavanya, V., Saranya, G., Lavanya, S.: Attributes based storage system for secure de-duplication of encrypt data in cloud. J. Comput. Theor. Nanosci. **17**(4), 1937–1942 (2020)
15. Gang, F., Wei, D.: Dynamic deduplication algorithm for cross-user duplicate data in hybrid cloud storage. Secur. Commun. Netw. (2022). https://doi.org/10.1155/2022/8354903
16. Lenin, J., Begum, B.R., Yousuf, T.S., Karthikeyan, R.: A secured storage scheme for cloud environment using ECC-IRNS based deduplication approach. In: 2022 International Conference on Inventive Computation Technologies (ICICT), pp. 901–911. IEEE (2022)
17. Yao, W., Hao, M., Hou, Y., Li, X.: FASR: an efficient feature-aware deduplication method in distributed storage systems. IEEE Access **10**, 15311–15321 (2022)
18. Verma, G.: Secure client-side deduplication scheme for cloud with dual trusted execution environment. IETE J. Res. (2022). https://doi.org/10.1080/03772063.2021.2017360
19. Ma, X., Yang, W., Zhu, Y., Bai, Z.: A secure and efficient data deduplication scheme with dynamic ownership management in cloud computing. In: 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), pp. 194–201. IEEE (2022)
20. Vignesh, R., Preethi, J.: Secure data deduplication system with efficient and reliable multi-key management in cloud storage. J. Internet Technol. **23**(4), 811–825 (2022)
21. Gnana Jeslin, J., Mohan Kumar, P.: Decentralized and privacy sensitive data de-duplication framework for convenient big data management in cloud backup systems. Symmetry **14**(7), 1392 (2022)
22. Athira, A.R., Sasikala, P., Reka, R.: An efficient secure data deduplication and portability in distributed cloud server using whirlpool-Hct and Lf-Wdo. Telematique **21**(1), 5078–5085 (2022)
23. Yang, Z., Li, J., Lee, P.P.: Secure and lightweight deduplicated storage via shielded {deduplication-before-encryption}. In: 2022 USENIX Annual Technical Conference (USENIX ATC 22), pp. 37–52 (2022)
24. Gupta, R., Saxena, D., Gupta, I., Makkar, A., Singh, A.K.: Quantum machine learning driven malicious user prediction for cloud network communications. IEEE Netw. Lett. **4**(4), 174–178 (2022)
25. Gupta, R., Gupta, I., Singh, A.K., Saxena, D., Lee, C.N.: An IoT-centric data protection method for preserving security and privacy in cloud. IEEE Syst. J. (2022). https://doi.org/10.1109/JSYST.2022.3218894
26. Gupta, R., Saxena, D., Gupta, I., Singh, A.K.: Differential and triphase adaptive learning-based privacy-preserving model for medical data in cloud environment. IEEE Netw. Lett. **4**(4), 217–221 (2022)
27. Gupta, R., Singh, A.K.: A differential approach for data and classification service-based privacy-preserving machine learning model in cloud environment. New Gener. Comput. **40**(3), 737–764 (2022)
28. Singh, A.K., Gupta, R.: A privacy-preserving model based on differential approach for sensitive data in cloud environment. Multimed. Tools Appl. **81**(23), 33127–33150 (2022)
29. Muthunagai, S.U., Anitha, R.: TDOPS: time series based deduplication and optimal data placement strategy for IIoT in cloud environment. J. Intell. Fuzzy Syst. **43**(1), 1583–1597 (2022)
30. Yang, X., Lu, R., Shao, J., Tang, X., Ghorbani, A.A.: Achieving efficient secure deduplication with user-denied access control in cloud. IEEE Trans. Dependable Secur. Comput. **19**(1), 591–606 (2022)
31. Li, J., Li, T., Liu, Z., Chen, X.: Secure deduplication system with active key update and its application in IoT. ACM Trans. Intell. Syst. Technol. **10**(6), 1–29 (2019)
32. Shin, H., Koo, D., Hur, J.: Secure and efficient hybrid data deduplication in edge computing. ACM Trans. Internet Technol. (2022). https://doi.org/10.1145/3537675
33. Gupta, A., Pandey, O.J., Shukla, M., Dadhich, A., Mathur, S., Ingle, A.: Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research, p. 14061015 (2014)
34. Gupta, A., Pandey, O.J., Shukla, M., Dadhich, A., Ingle, A., Gawande, P.: Towards context-aware smart mechatronics networks: integrating swarm intelligence and ambient intelligence. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), p. 14210992 (2014)
35. Siddique, M., Panda, D.: Prediction of stock index of Tata steel using hybrid machine learning based optimization techniques. Int. J. Recent Technol. Eng. **8**(2), 3186–3193 (2019)
36. Siddique, M., Panda, D.: A hybrid forecasting model for prediction of stock index of Tata motors using principal component analysis, support vector regression and particle swarm optimization. Int. J. Eng. Adv. Technol. **9**(1), 3032–3037 (2019)
37. Sarella, V.R., Prasad Reddy, P.V.G.D., Krishna Rao, S., Padala, P.: Efficient energy clustering adaptive routing procedure for wireless sensor networks. J. Glob. Inf. Manag. **25**(4), 125–138 (2017)

Mohd Akbar is currently working as a Lecturer at the University of Technology and Applied Sciences, Muscat, Sultanate of Oman. He is a Post Doctoral Fellow Scholar from Singapore Institute of Technology (SIT), Singapore. He holds a Ph. D. in Computer Science and Engineering and is a highly accomplished professional with a strong background in Computer Science and Engineering. He has published several research papers in Journals and Patents to his credit. His research areas of interest expertise Artificial Intelligence, Machine Learning, Deep Learning, Big Data Analytics, Cloud Computing, Blockchain, Internet of Things, Image Processing and Telehealth. He has more than two decades of teaching experience, in national and international, in higher education. In addition to the academic excellence, he has also been involved in content development for various Computer Science/IT related courses, research activities, editorial board members, reviews, technical committees, chief guest, keynote speakers, professional body members, and organizing the symposiums.

Irshad Ahmad is currently working as a Lecturer at the University of Technology and Applied Sciences, Muscat, Sultanate of Oman. He holds a Master in Computer Application and is a highly accomplished professional with a strong background in the Computer field. He has published research papers in Journals and Conferences. His research areas of interest expertise Artificial Intelligence, Machine Learning, Deep Learning, Big Data Analytics, Cloud Computing, Internet of Things and Image Processing. He has 17+ years of teaching experience, in national and international, in higher education. In addition to the academic excellence, he has also been involved in content development for various Computer Science/IT related courses, research activities, editorial board members, reviews, technical committees, professional body members, and organizing the symposiums.

Mohsina Mirza is currently working as a Lecturer at the Global College of Engineering and Technology, Muscat, Sultanate of Oman. She is a Ph. D. Scholar and well experienced, having more than 14 years of experience in the teaching sector in Oman and India. She is a hardworking and dedicated professional focused on engineering education, continuing her research on artificial intelligence/software defined networking/machine learning, and leading campus and professional service activities. She had been fortunate to be certified by esteemed organizations like Cisco (CCNA) and Microsoft (MTA), and NVIDIA. Her doctoral research is based on “intelligence-driven experiential network architecture for automatic routing in software-defined networking” wherein to quantify the security level of an SDN-MN, we define the NME that is used to derive the network security level. Calculating the NME regarding the dynamic factors in an SDN-MN is a multiple criteria decision-making problem. Given a chance. Her other areas of research include machine learning, artificial intelligence, cyber security, IOT, IIOT, etc.

Manavver Ali is currently working as a Lecturer at the University of Technology and Applied Sciences, Muscat, Sultanate of Oman. He holds a master’s in computer science and is a highly accomplished professional with a strong background in Computer Science. His research areas of interest expertise Artificial Intelligence, Machine Learning, Deep Learning, Big Data Analytics, Cloud Computing, Blockchain, Internet of Things, Image Processing and Telehealth. He has more than 15 years of teaching experience in international, in higher education.

Praveen Barmavatu is working as an Assistant Professor in the Department of Mechanical Engineering, Faculty of Engineering, Universidad Tecnológica Metropolitana, Santiago, Chile. He has more than 06 years of experience in the Teaching as Assistant Professor. He also served as Visiting Lecturer to other Institutions for Extension Lectures on various topics. He is one of the youngest doctorates in India and well known for his research contribution in the field of heat exchanging systems subjected to heavy industries and Machine Learning applications for the heat transfer. He has filed his inventions and published more than 06 patent grants. He holds more than 05 book chapters and over 34 papers in the reputed International scientific journals. He has participated and presented papers in many national and international conferences. He supervised 3 Ph.D. thesis and supervising many doctoral, master’s, and bachelor’s student theses. He is a regular reviewer for many topclass scientific journals and also has experience in funded projects.