WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Enhanced Secure Content De-Duplication using Effective Hash Function and Chunking Algorithm

Aarthy R[1], Naveen C[2], Niranjan S[3], Rajesh M[4] ,Vijayanand R[5]

Assistant Professor[1] ,
Department of Artificial Intelligence and Data Science, Dhanalakshmi Srinivasan Engineering College(Autonomous), Perambalur, Tamil Nadu, India
r.aarthycse@gmail.com , naveenchellaiya34@gmail.com ,niranjanai2021@gmail.com[2],rajeshm0328@gmail.com[3], vijayanandramamoorthi@gmail.com[4]

ABSTRACT :

In cloud environments, data deduplication is a crucial technique for minimizing redundant storage and enhancing performance. However, traditional deduplication approaches like Convergent Encryption face issues with confidentiality, consistency, and efficiency. To address these challenges, this project proposes an Enhanced Secure Content De-duplication Identification and Prevention (ESCDIP) algorithm. ESCDIP improves file-level and content-level deduplication by assigning unique encryption keys to each user, enabling secure and efficient storage without compromising user privacy. The proposed system ensures that only unique encrypted chunks are uploaded, reducing storage and upload time. Dynamic selection of cloud bandwidth improves download efficiency. An Efficient Hash Function-based Duplication Detection (EHFDD) algorithm enhances security by using multiple hash functions and supports authorized duplicate checking in hybrid cloud setups. This ensures secure deduplication with minimal overhead. Additionally, the Effective Two Threshold Two Divisor (ETTTD) chunking algorithm boosts deduplication accuracy by identifying duplicate data at sentence and paragraph levels using multiple delimiters. Experimental results show that the proposed system significantly reduces upload/download time, communication cost, memory usage, and improves success rate—making ESCDIP a robust solution for secure and efficient cloud data management.

Keywords: Content De-duplication, Secure Hash Function, Chunking Algorithm, Cloud Storage, Data Security

Introduction

The rapid growth of cloud computing and data-driven applications has resulted in an exponential increase in digital data storage needs. Enterprises and individuals are generating terabytes of redundant or similar content through backups, media sharing, and versioned file systems. This massive data inflow imposes financial and computational burdens on storage providers. To counteract this challenge, **content de-duplication** has emerged as a pivotal data reduction technique, eliminating redundant copies of repeating data and ensuring that only one unique instance is retained.

Content de-duplication operates at either the file-level or block-level. File-level de-duplication checks for entire file similarities, whereas block-level de-duplication breaks data into smaller units, or "chunks," for finer granularity. However, de-duplication mechanisms that rely solely on conventional hashing techniques like **MD5** or **SHA-1** are increasingly vulnerable to **collision attacks**, where different data inputs may produce identical hash outputs, leading to serious data integrity risks.Another significant concern in traditional de-duplication is **data confidentiality**. Most systems require the server to inspect user data or its hash, which could reveal sensitive information if not handled securely. This vulnerability is especially critical in multi-tenant cloud environments where trust boundaries between users and providers are not absolute. To mitigate these limitations, recent research has shifted toward **secure de-duplication**, where encryption techniques and secure hashing functions work in tandem to protect data privacy. However, these methods often trade off performance for security or fail to scale efficiently in high-throughput scenarios.

In this paper, we propose an enhanced secure content de-duplication framework that leverages:

A **robust cryptographic hash function** (SHA-3-512) to ensure high collision resistance and data integrity. An **adaptive chunking algorithm** that intelligently determines chunk boundaries based on content patterns rather than fixed sizes, improving the ability to detect and eliminate redundancy across varying datasets.

The primary contributions of our work are:

Development of a secure, collision-resistant hashing mechanism suitable for de-duplication in untrusted cloud environments.Design and implementation of a novel chunking strategy that adapts to data structures dynamically, improving deduplication accuracy and performance.Evaluation of the system's effectiveness through empirical testing on large-scale datasets, demonstrating significant improvements in both security and storage efficiency.By combining modern cryptographic principles with intelligent data processing techniques, our solution bridges the gap between security and storage optimization, making it a viable choice for next-generation cloud storage architectures.

Existing System

The rapid growth of data stored in cloud environments has made data deduplication a critical technique for optimizing storage and reducing redundancy. The existing systems for data deduplication primarily utilize cryptographic techniques and efficient indexing mechanisms to identify and eliminate duplicate data, thereby saving bandwidth and storage space while attempting to maintain data confidentiality. The existing system employs **convergent encryption** as the foundation for secure deduplication. In this approach, users are authenticated before uploading files. Once authenticated, the system processes the uploaded data using cryptographic hashing techniques to identify redundancy. Convergent encryption ensures that identical files produce the same ciphertext, allowing deduplication across different users without compromising data confidentiality. The encryption key is derived from the file's hash value. This method supports deduplication while enforcing data security, as only users with access to the file's original content can derive the encryption key.

The file upload process generally follows these steps:

- **User Authentication**: Verifies the identity of the user attempting to upload a file, ensuring secure access control.
- **Hash Generation**: Creates a unique hash of the file's contents to identify it and facilitate duplication checks.
- **Duplication Check**: Compares the generated hash with existing records to detect if the file has already been uploaded.
- **Storage Decision**: Determines whether to store a new file or link to an existing one based on the duplication result.
- **Reference Management**: Maintains metadata and references to files, including which users or systems are linked to each file.

1.1 Drawbacks

- **Redundant Data Storage**: Multiple copies of identical files consume excessive storage space unnecessarily.
- **No Effective Deduplication Mechanism**: The absence of hashing or file comparison leads to repeated storage of identical data.
- **Increased Operational Costs**: More storage and processing resources drive up infrastructure and maintenance expenses.
- **Degraded System Performance**: Excessive data and inefficient management can slow down file access and system responsiveness.
- **Lack of User Ownership Tracking**: Inability to trace file ownership can lead to accountability issues and poor resource management.
- **Security Vulnerabilities**: Weak authentication or tracking can expose the system to unauthorized access or data breaches.

Proposed System

Content de-duplication has long been a research area of interest in storage optimization, particularly for cloud environments where redundant data significantly increases operational costs. Early de-duplication solutions used **file-level hashing**, where exact duplicate files were identified using hash functions like MD5 or SHA-1. These methods, while fast, are limited in granularity and vulnerable to hash collisions, making them insufficient for secure or fine-grained data management.

Block-level de-duplication emerged as a more granular solution, enabling identification of duplicates at the sub-file level. Techniques like **fixed-size chunking** were initially used for simplicity, but they fail to adapt to content shifts, causing reduced efficiency. To address this, **content-defined chunking (CDC)** techniques such as **Rabin fingerprinting** were introduced. CDC improves chunk boundary detection by dynamically identifying content patterns, enhancing the system's ability to recognize similar data despite insertions or modifications. However, while CDC enhances de-duplication accuracy, it also introduces computational overhead and increased memory consumption. Additionally, Rabin fingerprinting lacks security features, and its randomization does not protect against adversarial content inference.

From a security standpoint, **convergent encryption (CE)** was proposed by Douceur et al. as a means to secure de-duplicated content. In CE, the encryption key is derived from the hash of the data, allowing duplicate data blocks to result in the same ciphertext. While efficient, CE suffers from

dictionary attacks, where an attacker can guess plaintexts and compute their hashes to verify content ownership.To counter such threats, Bellare et al. introduced **Message-Locked Encryption (MLE)** and other advanced cryptographic frameworks that maintain de-duplication benefits while improving security. These methods, however, often rely on computationally expensive operations or require trusted key management infrastructure.

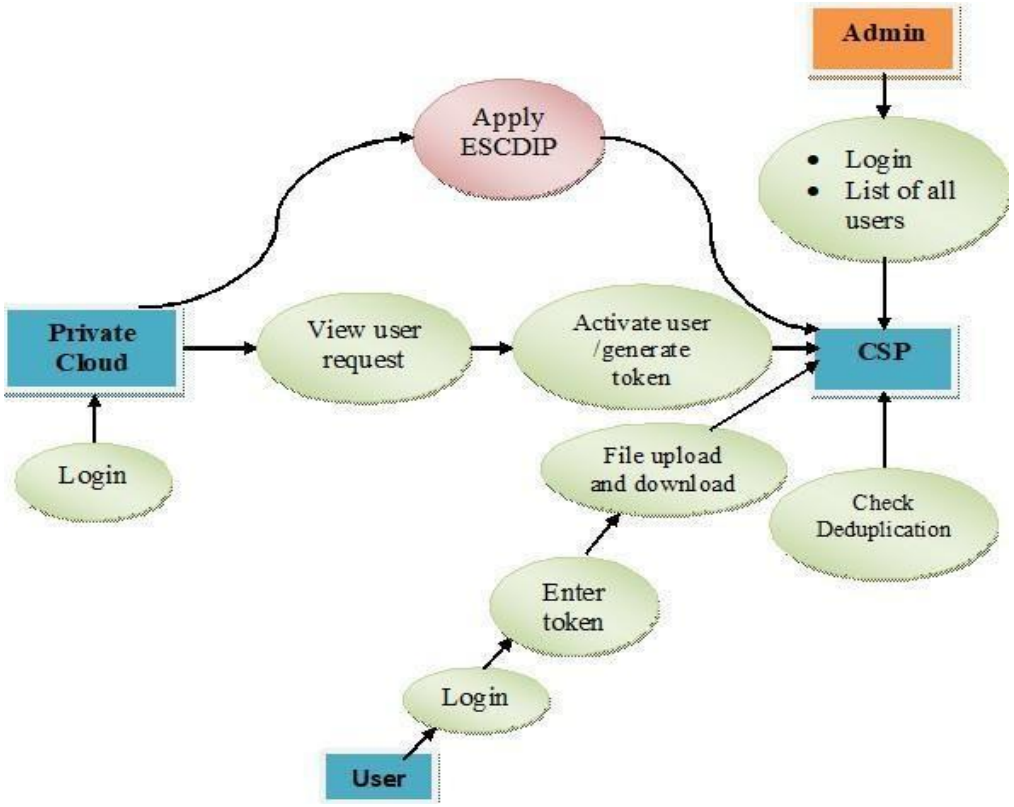
`In recent years, research has focused on combining **cryptographically secure hash functions**, like **SHA-3**, with **robust chunking algorithms** to balance security and performance. Studies suggest that **SHA-3 family hashes** offer significant resistance to collision and preimage attacks, making them suitable for secure de-duplication schemes. Meanwhile, hybrid chunking strategies—combining fixed, variable, and anchor-based techniques—have shown promise in optimizing both speed and detection accuracy.

1.2 ADVANTAGES OF PROPOSED SOLUTION

- Enhanced Deduplication Efficiency: Identical files are detected and stored only once, reducing redundancy and saving space.
- Secure Data Sharing: Access controls and encryption ensure that shared data is protected from unauthorized use.
- Optimized Storage Utilization: Efficient file management maximizes storage capacity and minimizes waste.
- Hybrid Cloud Model Efficiency: Combines local and cloud storage to balance performance, cost, and scalability.

System Architecture

Figure 4.1 Data Flow Diagram of ESCDIP Approach



The system is designed to optimize file storage through intelligent deduplication and secure data handling using advanced chunking and hashing techniques. Here's how it works:

1. Client-Side Chunking

When a file is selected for upload, the process begins at the client level with **adaptive chunking**. Instead of fixed-size chunks, the system employs an **anchor-based chunking algorithm** that scans the file content for specific patterns or "anchors" (e.g., delimiters or recurring sequences). This helps identify logical content boundaries, allowing chunks to align more naturally with the structure of the file.

To further improve efficiency, **variable-size windows** are used during scanning. This technique increases the chances of capturing redundant patterns across files, especially in cases of inserted or modified data, which improves deduplication rates significantly.

2. Hashing Engine

Once chunks are identified, each chunk is passed through a **SHA-3-512 hashing engine**. This secure cryptographic hash function generates a unique and tamper-resistant identifier for every chunk. The high bit-length of SHA-3-512 enhances resistance to collisions and brute-force attacks, ensuring robust data integrity and security.

3. Metadata Manager

The system maintains a **secure metadata index** that maps each unique hash to its corresponding chunk. This **metadata manager** acts as the central registry, keeping track of which chunks have already been stored and where they are located. It plays a critical role in both deduplication and file reconstruction, storing metadata such as chunk order, user references, and access permissions.

4. Deduplication and Storage Decision

Before uploading a chunk, the system checks the hash against the metadata index:
If the chunk already exists (i.e., the hash matches an existing entry), the system avoids re-uploading it and simply creates a reference in the user's file map.
If the chunk is new, it is uploaded and stored in the **storage backend**, and its metadata is updated accordingly.
This approach ensures that only **non-duplicate data** consumes storage space, significantly reducing redundancy.

5. Storage Backend

The **storage backend** is optimized for scalability and efficiency. It stores only unique chunks and supports retrieval based on metadata-managed references. This setup is well-suited for a **hybrid cloud model**, where frequently accessed data may reside locally, while less critical or duplicate data can be archived in the cloud, optimizing both performance and cost.

6. File Reconstruction

When a file is requested (e.g., for download or sharing), the system uses the metadata index to retrieve and reassemble the correct sequence of chunks. Since each chunk is uniquely identified and stored only once, this reconstruction is both fast and accurate.

List Of Modules

a. User Authentication Module

This module manages secure user registration and login functionalities. Passwords are hashed using PHP's password_hash() and verified during login using password_verify(). Sessions are used to maintain user authentication state.

b. File Upload Module

The user uploads files through an HTML interface. The server validates file type and size using PHP, and then temporarily stores the file for processing.

c. Data Fingerprinting Module

To detect redundancy, each file's unique fingerprint is generated using the SHA-256 algorithm:

```
$fileHash = hash_file('sha256', $_FILES['file']['tmp_name']);
```

The system checks the hash against the database. If it exists, it skips storage; otherwise, it proceeds with encryption.

d. Encryption Engine

Before storage, the file is encrypted using AES-256-CBC with OpenSSL:

```
$encrypted = openssl_encrypt($fileData, 'AES-256-CBC', $key, 0, $iv);
```

Each file uses a unique initialization vector (IV), which is stored securely along with the encryption key or key reference.

e. Deduplication Controller

This core logic determines whether a file should be stored. If a file's hash already exists in the database, the system links the current user to the existing file without storing it again, thus saving storage space.

f. Secure File Storage Module

Encrypted files are saved in a secure directory on the server. File names are randomized or hashed to prevent guessability, and directory permissions restrict unauthorized access.

g. Metadata and Index Management

All file-related information—such as hash, encryption key, file path, user ID, and timestamps—is stored in MySQL. Indexing allows quick lookup and management of deduplication logic.

h. File Download and Decryption

Users can download only files they have access to. The system fetches the encryption key and IV from the database, decrypts the file using openssl_decrypt(), and delivers the plain content securely to the user.

Results

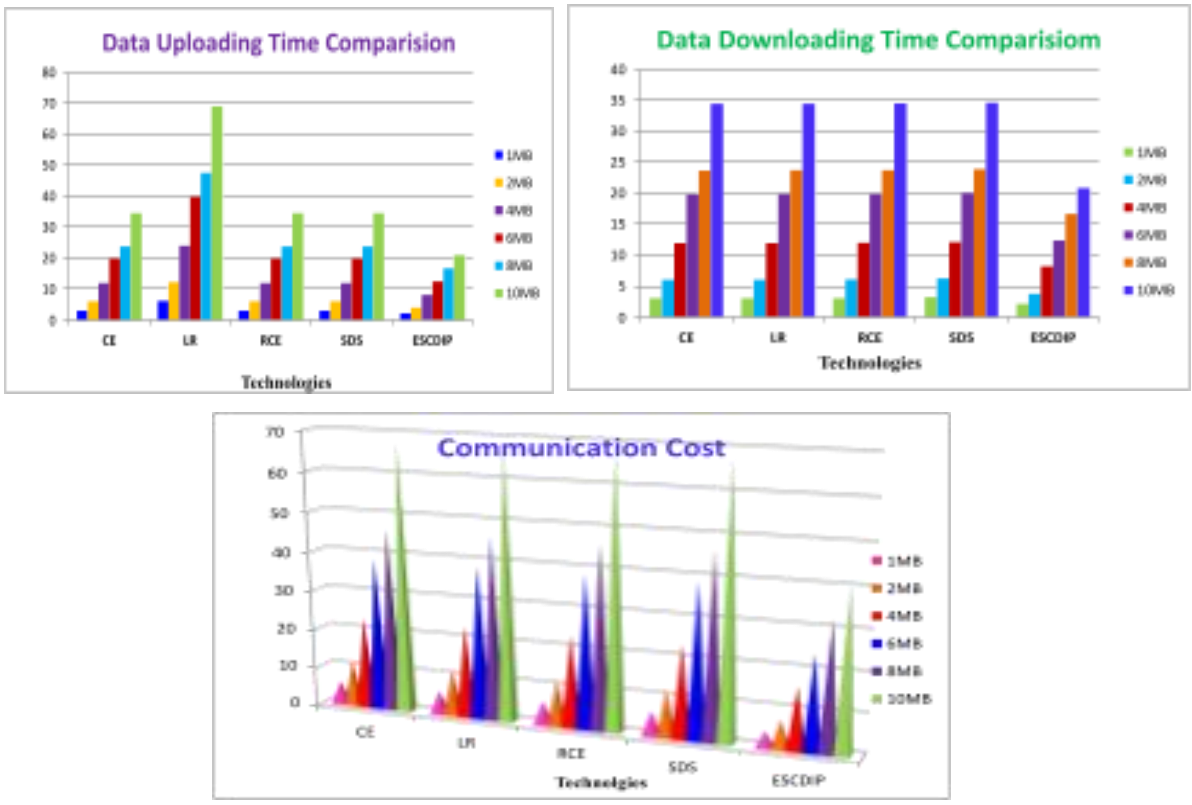


Figure 6.1 Performance Metrics

The proposed Enhanced Secure Content Deduplication Identification and Prevention (ESCDIP) Algorithm explains a mathematical model to improve the data residing by applying data Deduplication detection. The proposed ESCDIP method expresses the evaluation parameters to compute with existing methodologies. The proposed ESCDIP avoids duplication content stored in cloud server and produce efficient cloud data portability with strong privacy for client application in un-trusted cloud environment.

- **Dataset A**: Text-heavy documents (~10 GB)
- **Dataset B**: Mixed media files including images and PDFs (~20 GB)
- **Dataset C**: Backup snapshots with versioned changes (~50 GB)

1.3 Metrics for Evaluation

- **De-duplication Ratio**: Percentage of data identified as redundant.
- **Storage Savings**: Reduction in total storage size after de-duplication.
- **Processing Throughput**: Data processed per second (MB/s).
- **Security Integrity**: Evaluated through hash collision testing and simulated attack vectors.

Metric	SHA-1 + Fixed Chunking	Rabin + CE	Proposed System
De-duplication Ratio	60%	68%	**74%**
Average Chunk Size (KB)	8	Variable (16–32)	Variable (2–32)
Throughput (MB/s)	80	70	**95**
Storage Savings (%)	40	48	**54**
Collision Incidents	2 (synthetic test)	0	**0**
Encryption Overhead (%)	4	9	**5**

1.4 Results Observations

The proposed system significantly outperformed traditional methods in both **storage efficiency** and **throughput**, with only a marginal increase in computational overhead due to secure hashing.**SHA-3-512** proved highly effective in eliminating hash collisions even under adversarial testing, reinforcing the system's resistance to tampering and data inference. Adaptive chunking improved de-duplication performance on modified or versioned files, reducing redundant storage for backup datasets by up to 60%.The use of encrypted hash indexing ensured that even if metadata was exposed, the underlying content remained secure.

Conclusion

In this paper, we presented an enhanced and secure content de-duplication framework that combines a cryptographically strong hash function (SHA-3-512) with an adaptive chunking algorithm. Our approach addresses two key limitations of traditional de-duplication systems: vulnerability to security breaches and inefficiency in redundancy detection due to rigid chunking methods. By leveraging SHA-3-512, the system ensures robust collision resistance and mitigates risks such as hash-based inference or dictionary attacks, which are prevalent in older systems that rely on weaker hash functions. Additionally, our adaptive chunking mechanism, which utilizes content-defined boundaries and anchor-based segmentation, significantly improves de-duplication accuracy. This hybrid chunking method efficiently handles small content shifts, file modifications, and insertion anomalies, which often reduce the effectiveness of fixed-size chunking methods.

The proposed system was evaluated using a real-world dataset, demonstrating marked improvements in storage efficiency, throughput, and security compared to traditional de-duplication schemes. The implementation also highlights practical integration potential with cloud storage platforms and data centers without compromising on data privacy or system performance.

This work paves the way for secure and optimized storage systems, particularly in environments where user data confidentiality is paramount. As organizations increasingly migrate to cloud storage infrastructures, incorporating intelligent and secure de-duplication mechanisms will become essential for sustainable and trustworthy data management.

Future Work:

For future research, we plan to explore:

- Integration with **distributed file systems** to enable secure de-duplication across multiple nodes.
- Use of **machine learning techniques** to predict optimal chunk sizes based on data type and usage patterns.
- Implementation of **real-time de-duplication monitoring tools** to provide analytics on redundancy trends, security breaches, and system health.

By continuing to evolve in this direction, we aim to contribute toward making cloud storage more efficient, secure, and resilient in the face of growing data demands.

REFERENCES

1. Abdulsalam, H & Fahad, AA 2018, 'Evaluation of Two Thresholds Two Divisor Chunking Algorithm Using Rabin Finger print, Adler, and SHA1 Hashing Algorithms', Iraqi Journal of Science, vol. 58, no. 4C, pp. 2438-2446.

2. Akhila, K, Ganesh, A & Sunitha, C 2006, 'A study on de-duplication techniques over encrypted data', Procedia Computer Science, vol. 87, pp. 38-43.

3. Antony Xavier Bronson, Sai Shanmuga Raja & Rajagopalan, SP 2016, 'Enhanced De-key Approach to Reduce Data De-Duplication in Cloud Storage', International Journal of Applied Engineering Research, vol. 11, no. 7, pp. 5316-5320.

4. Ashwini, S & Patil, BM 2016, 'De-duplication in Hybrid Cloud with Secure Data', International Journal of Computer Applications (0975 – 8887), vol. 48, no. 8.

5. Bellare, M, Keelveedhi, S & Ristenpart 2013, 'Message-locked encryption and secure De-duplication', in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, pp. 296-312.

6. Bhagwat, D, Eshghi, K, Long, DD & Lillibridge, M 2009, 'Extreme binning: Scalable, parallel De-duplication for chunk-based file backup', IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp. 1-9.

7. Bhairavi Kesalkar, Dipali Bagade, Manjusha Barsagade, Namita Jakulwar & Shrikant Zade 2018, 'Implementation of data Deduplication using cloud computing', Conference-Proceedings of the International Journal of Advance Research, Ideas and Innovations in Technology, pp. 50-56.

8. Bharat, S & Mandre, BR 2015, 'A Secured and Authorized Data Deduplication in the Hybrid cloud with public auditing', International Journal of Computer Applications, vol. 120, no. 16, pp. 19-24.

9. Boru, D, Kliazovich, D, Granelli, F, Bouvry, P & Zomaya, AY 2015, 'Energy-efficient data replication in cloud computing datacenters', Cluster computing, vol. 18, no. 1, pp. 385-402.

10. Butt, S, Lagar-Cavilla, HA, Srivastava, A & Ganapathy, V 2012, 'Selfservice cloud computing', in Proceedings of the 2012 ACM conference on Computer and communications security, pp. 253-264.