ENHANCING CLOUD DATA SECURITY AND STORAGE THROUGH DEDUPLICATION AND AES

A
Project Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

G. Vanshika 1602-21-737-060

M. Jaya Ankitha 1602-21-737-021

Under the guidance of

Mrs. B. Leelavathy

Assistant Professor

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh,

Hyderabad-500031

2025

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **G. Vanshika** and **M. Jaya Ankitha** bearing hall ticket number **1602-21-737-060** and **1602-21-737-021** hereby declare that the project report entitled **Enhancing Cloud Data Security and Storage Through Deduplication and AES** under the guidance of **Mrs. B. Leelavathy, Assistant Professor,** Department of Information Technology, Vasavi College of Engineering, Hyderabad is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology.**

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

G. Vanshika
1602-21-737-060

M. Jaya Ankitha
1602-21-737-021

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Enhancing Cloud Data Security and Storage Efficiency through Deduplication and AES** being submitted by **G. Vanshika** and **M. Jaya Ankitha** bearing **1602-21-737-060** and **1602-21-737-021** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology in a record of bonafide work carried out by them under my guidance.

Mrs. B. Leelavathy
Assistant Professor
Internal Guide

Dr. K. Ram Mohan Rao
Professor & HOD, IT

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of the Main project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

It is with immense pleasure that we would like to take the opportunity to express our humble gratitude to **Mrs. B. Leelavathy, Assistant Professor, Department of Information Technology** under whom we executed this project. We are also grateful to **Mrs. C. Sireesha, Assistant Professor, Department of Information Technology,** for her guidance. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

We are very much thankful to **Dr. K. Ram Mohan Rao, Professor & HOD, Department of Information Technology,** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. S. V. Ramana**, **Principal** of **Vasavi College of Engineering** for providing facilities. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time

ABSTRACT

The exponential growth of digital data, cloud storage systems face significant challenges in managing storage efficiency while ensuring data security. Data deduplication, a technique that eliminates redundant copies of data, plays a crucial role in optimizing storage costs and reducing redundant data. However, traditional deduplication mechanisms are vulnerable to security threats, like data identifier manipulation and attacks based on traffic analysis. This project presents a secure deduplication framework that integrates block level deduplication while leveraging Advanced Encryption Standard (AES) to enhance data confidentiality. Our approach employs a deduplication check using hashing and makes sure to ensure that inter-user deduplication remains undetectable, preventing unauthorized access and potential exploitation by malicious clients. By enforcing cryptographic controls and maintaining the integrity of file-to-block mappings, the proposed system mitigates security risks while preserving storage efficiency. Experimental evaluations demonstrate that our method significantly reduces storage space, while introducing only minimal computational overhead due to encryption. The results highlight the feasibility of achieving both security and effective deduplication, making our approach a promising solution for secure and efficient cloud storage management.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AES	Advanced Encryption Standard
ABE	Attribute-Based Encryption
CSP	Cloud Service Provider
SHA	Secure Hash Algorithm
CE	Convergent Encryption
FTPS	File Transfer Protocol Secure
HMAC	Keyed-Hash Message Authentication Code

1. INTRODUCTION

In today's digital era, data is being generated at a striking rate across individuals, enterprises, and institutions. This surge in data has placed immense demand on cloud storage systems to not only store vast volumes of information but to do so efficiently, securely, and cost-effectively. Cloud computing has emerged as the dominant paradigm for data storage and access, offering scalability and remote accessibility. However, with this growth comes the challenge of storing redundant and duplicate data, which significantly inflates storage costs and introduces inefficiencies.

To overcome this, data deduplication has become a widely adopted technique. It works by eliminating repeated data blocks, ensuring that only unique data is stored while duplicates are replaced with references. While deduplication greatly reduces storage space, it poses serious security challenges, especially when combined with encrypted content. Traditional encryption schemes often break deduplication because the same content encrypted with different keys results in different cipher texts.

This project proposes a secure deduplication framework that addresses both efficiency and confidentiality. By integrating block-level deduplication with AES-256 encryption and SHA-256 hashing, the system ensures that data remains protected even while optimizing storage.

1.1 Problem Statement – Overview

With the rapid expansion of cloud storage usage and growing volumes of digital data, cloud storage providers face exponential data growth, leading to high storage, increased storage costs, and inefficient resource utilization due to redundant files being stored multiple times by different users. To address this, data deduplication has emerged as a key technique that removes redundant data by ensuring only a single instance of identical data is stored while references are maintained for duplicate copies.

However, existing deduplication methods come with critical security vulnerabilities like Identifier Manipulation Attacks etc., making them susceptible to exploitation by malicious users. Traditional file-level deduplication identifies duplicate files across users but fails to detect redundancy within a file. There is a clear need for a

secure and efficient deduplication system that ensures storage optimization while preserving security and encrypts data before storage without breaking deduplication and preventing unauthorized access.

By encrypting only unique blocks and managing metadata securely, the system supports both storage optimization and confidentiality. The proposed framework ensures that deduplication can coexist with encryption, enabling secure and efficient cloud storage suitable for real-world multi-user environments.

This system not only addresses the current limitations of deduplication security but also establishes a scalable foundation for handling large datasets in the cloud.

1.2. Motivation

The motivation behind this project lies in addressing the dual challenge faced by cloud storage systems: optimizing storage space while maintaining strict security controls. The demand for secure cloud storage is rapidly increasing due to the growing reliance on cloud-based services for personal, organizational, and government data storage.

Numerous real-world applications and scenarios emphasize the need for efficient deduplication combined with strong encryption:

- **Enterprise Data Management:** Enterprises dealing with vast quantities of files for backups, archiving, and content sharing require storage solutions that are both secure and cost-effective.
- **Personal Cloud Storage:** Users increasingly store personal data, including photos, documents, and videos, which often contain redundant files, increasing storage costs.

Addressing these requirements necessitates a secure deduplication framework that incorporates modern encryption techniques like AES-256 and efficient hashing algorithms like SHA-256, ensuring both space optimization and uncompromised data security.

1.3. Scope & Objectives of the Proposed Work

Scope:

The scope of this project extends to the design, development, and validation of a comprehensive secure cloud storage system that combines the advantages of deduplication and encryption to optimize both storage and security. It focuses on implementing a robust block-level deduplication mechanism integrated with AES-256 encryption and SHA-256 hashing. The system targets both intra-user and inter-user deduplication, ensuring data integrity and confidentiality during storage and retrieval operations in a cloud environment.

This project is particularly significant in contemporary times, where cloud storage services are extensively used by businesses, individuals, and institutions. Reducing storage costs without compromising security is a key requirement for scalable cloud storage infrastructures. The proposed solution directly addresses these demands by ensuring that redundant data is identified and removed at the block level, while maintaining the confidentiality of the data through strong encryption.

The scope includes the development of a cloud storage framework using Java-based technologies such as Servlets, JSP, and JDBC, integrated with cloud services like Drive HQ. Additionally, the project involves setting up user authentication, encryption key management, metadata management, and secure file reconstruction mechanisms.

Objectives:

- **Implementing Block-Level Deduplication:**
    - Splitting files into uniform 1KB blocks.
    - Generating SHA-256 hashes for each block.
    - Identifying and eliminating redundant blocks across multiple users.
- **Integrating AES-256 Encryption:**
    - Encrypting only unique data blocks to reduce encryption overhead.
    - Dynamically generating secure encryption keys.
    - Protecting encrypted data blocks against unauthorized access.
- **Storing Data Securely in the Cloud:**
    - Uploading encrypted blocks to a cloud storage service.

- Ensuring metadata (hash values, block references, encryption keys) is securely maintained in a relational database.

- **User Authentication and Access Control:**
  - Implementing secure login mechanisms.
  - Issuing unique secret keys for file download authorization.
  - Restricting access to authorized users only.

This multi-objective approach ensures the proposed system is not only secure and efficient but also scalable and adaptable to real-world applications involving massive and sensitive datasets.

1.4. Organization of the Report

This report provides a structured and detailed overview of the research, design, and implementation of a secure cloud storage system that integrates block-level deduplication with AES-256 encryption. The document flows logically from problem identification to performance evaluation, highlighting how data redundancy and security challenges are tackled simultaneously. The specific sections and order can vary depending on the report's purpose and audience.

Chapter-Wise Overview:

Chapter 1. Introduction: This chapter introduces the need for secure and optimized cloud storage. It outlines the challenges of redundant data storage, explains the importance of deduplication, and emphasizes the role of cryptographic techniques in securing user data.

Chapter 2. Literature Survey: This section reviews existing deduplication techniques and security mechanisms. It highlights the evolution from traditional file-level deduplication to secure block-level approaches, and identifies the limitations in current methods, especially related to performance and data confidentiality.

Chapter 3. Proposed System: This chapter presents the architecture and methodology of the system. It details how files are split, hashed, deduplicated, encrypted, and stored in the cloud. Functional modules such as authentication, encryption, cloud upload, and metadata management are thoroughly described.

Chapter 4. Experimental Setup and Results: This chapter explains the implementation setup, system specifications, and the testing environment. It includes a detailed analysis of results based on performance metrics such as storage savings, encryption efficiency, and hash time. Graphs and tables support the findings.

Chapter 5. Conclusion and Future Scope: This chapter summarizes the outcomes of the system, affirming that the objectives were met. It also discusses potential areas of improvement, including integrating CP-ABE, support for multimedia file types, and enhanced metadata protection mechanism.

2. LITERATURE SURVEY

Dong Zheng et al. [1] In their paper, a Secure data de-duplication and recovery scheme based on PEKS is constructed by using the matching relationship between keyword and trapdoor of searchable encryption to achieve file equality test in ciphertext state and using proxy re-encryption to achieve data recovery. But there are still some shortcomings, such as the current scheme of this paper only supports equality test at the file level.

Nagappan Mageshkumar et al. [2] Here the approach combines the Diffie-Hellman algorithm and symmetrical external decision to protect and popularize information, ensuring end-to-end encryption to encourage user adoption of cloud storage. The primary focus of this work is to examine the implementation of an enhanced cryptosystem to provide safe data protection, specifically by mitigating the presence of duplicated data. The proposed model achieves the space saving analysis more than of 3 % for ECC and 4 % for RSA approaches.

R. Aishwarya et al. [3] The proposed approach in the paper varies from traditional de-duplication systems, where each client has their own rights to the data that they have supplied. In the proposed hybrid cloud architecture, the S-CSP is hosted in the public cloud. According to a security study, the proposed research plan is reliable in terms of data protection. Eventually, this research study concludes that the digital storage security remains crucial, as well as the fact that many study questions are still unsolved.

Zhang Liang et al. [4] The paper employs the public bidding data for performance analysis. The algorithm proposed in the paper has an accuracy of 86.49%.The results show that the feature representation method based on named entities can effectively describe the text data and improve text information deduplication accuracy. The method in this paper is mainly aimed at the webpage data extracted from the text, which effectively solves the problem of low value density of the data obtained by the competitive intelligence information platform.

Pritesh Kumar Prajapati et al. [5] In this paper, various approaches for providing secure deduplication in cloud storage are discussed. The different schemes proposed

above can be employed with existing cloud storage services. Attribute Based Encryption (ABE) and Identity Based Encryption (IBE) have been researched to address client's security concerns and this paper does a literature review on such various proposed approaches for secure deduplication techniques in cloud storage.

Ojaghi Behnam et al. [6] In the paper, their primary focus lies in enhancing the QoS of V2X applications utilizing Open Radio Access Network (O-RAN) architecture. To this end, we propose a novel optimization framework that creates dynamic RAN slicing covering tailored functional split selection per slice, and Multi-access Edge Computing (MEC) server placement within O-RAN architecture.

Sunil B et al. [7] Their paper introduced a novel method implemented in Java using Cloud Sim (Cloud Simulator), which integrates Convergent Encryption (CE) and Modified Elliptic Curve Cryptography (MECC) algorithms. The primary objective of this approach is to develop secure data deduplication systems that simultaneously minimize data redundancy and ensure robust encryption in cloud storage environments.

Shilin Liu et al. [8] Their paper presented an enhanced AES algorithm based on key operation. The traditional Sub Bytes operation in AES replacing the current state matrix with values obtained from the S-box. and the Shift Rows is that every row of the state matrix is rotated to the left according to the fixed bit.

Umamaheswari S et al. [9] In their paper, the implementation of a crypto-processor based on AES and HMAC to address the inefficiencies of software-based encryption and authentication. Their implementation uses AES for encryption and HMAC for integrity and authentication. By implementing these functions in hardware, we achieve greater efficiency and security, while reducing resource consumption and power usage.

Deepa M et al. [10] In their work, a novel method for implementation of Substitution BOX using NAND gates is presented and the effectiveness of hardware implementation and its power requirements are presented. This implementation of Advanced encryption standard is carried out with Cadence Virtuoso. The implementation of AES algorithm is being carried out in secure file transfer protocols like FTPS, HTTPS.

Xixun Yu et al. [11] In their paper, they proposed a verifiable deduplication scheme called VeriDedup to address the above problems. It can guarantee the correctness of duplication check and support flexible tag generation for integrity check over encrypted data deduplication in an integrative way. Concretely, we propose a novel Tag-flexible Deduplication-supported Integrity Check Protocol (TDICP) based on Private Information Retrieval (PIR).

Junqi Chen et al. [12] The paper proposed a secure text cloud-edge collaborative fault-tolerant storage scheme and its data writing optimization method. Precisely, we first propose a Hierarchical text Cloud-Edge Collaborative Fault-Tolerant Storage (HCEFT) model to achieve system robustness, low edge storage overhead, and edge data privacy security.

Ayano Nakshima et al. [13] The research presents a new Advanced Encryption Standard (AES) S-Box hardware design based on linear mappings optimized by combining multiplicative and exponential offsets. We apply the offset methods to another S-Box based on the redundant Galois field arithmetic and evaluate the performance by logic synthesis.

Ishu Gupta et al. [14] This article presented a comparative and systematic study, and in-depth analysis of leading techniques for secure sharing and protecting the data in the cloud environment. The discussion about each dedicated technique includes: functioning for protecting the data, potential and revolutionary solutions in the domain, the core and adequate information including workflow, achievements, scope, gaps, future directions, etc. about each solution.

Wenbin Yao et al. [15] The given paper proposed a new Feature-Aware Stateful Routing method (FASR), aiming to reduce the system overhead and keep a high deduplication ratio in the distributed environment. Firstly, we design a feature-aware nodes selection strategy to choose similar nodes by extracting data feature and data distribution characteristics. This strategy will save the similarity calculation with the nodes that are not similar to the data.

Kumar P et al. [16] We found that whole-file deduplication achieves about three quarters of the space savings of the most aggressive deduplication for storage of live file systems, and 87% of the savings for backup images. We also studied file

fragmentation finding that it is not prevalent, and updated prior file system metadata studies, finding that the distribution of file sizes continues to skew toward very large unstructured files.

Cao Chun-Hua et al. [17] studied the automatic de-duplication method of software quality inspection data based on density-based spatial clustering of applications with noise (DBSCAN) clustering. Intelligent optimization algorithm is used to generate software quality inspection data by initializing individuals, calculating fitness function value.

An Sangwoo et al. [18] In their method cloud storage refers to scalable and elastic storage capabilities delivered as a service using Internet technologies with elastic provisioning and use based pricing that doesn't penalize users for changing their storage consumption without notice.

Kim Youngbeom et al. [19] Within this paper they give an overview of existing file storage services and examine Dropbox, an advanced file storage solution, in depth. We analyze the Dropbox client software as well as its transmission protocol, show weaknesses and outline possible attack vectors against users. Based on our results we show that Dropbox is used to store copyright-protected files from a popular file sharing network.

Maghsoudloo Mohammad et al. [20] In their case they proposed an SLA-aware multi-criteria data placement strategy in cloud storage, optimizing performance while meeting service-level agreements. Data deduplication also improves users experience by saving network bandwidth and reducing backup time when the clients perform the deduplication before uploading data to the storage server

Heqing et al. [21] In their paper they target the attacks from malicious clients that are based on the manipulation of data identifiers and those based on backup time and network traffic observation.

Vikas et al. [22] Their paper explored the significance of deduplication ratios related to specific capacity optimization techniques within the context of information lifecycle management.

Shen Wenting et al. [23] Proposed a secure cloud-edge collaborative fault-tolerant storage scheme that optimizes data writing while ensuring data availability and security in cloud environments.

Mhaisen Naram et al. [24] This review paper introduces a hybrid cloud storage system with a multilayer cryptosystem for secure deduplication, enhancing privacy and storage efficiency.

Joe C et al. [25] Designed a cloud secure storage mechanism using data dispersion and encryption to strengthen data confidentiality and integrity. Their method optimizes data distribution while meeting service-level agreements, ensuring efficient performance and reliability in cloud storage management.

Liu Mingyu et al. [26] This analysis reveals differences in the frequency response of the three different types of data storage systems and present a lightweight cloud storage auditing method with deduplication that ensures strong privacy protection.

P. Meye et al. [27] This research summarizes the architecture of cloud storage along with addressing data deduplication issues in cloud storage using hashing and MD5 techniques, aiming to optimize storage space and retrieval efficiency. Their study analyzes existing challenges, evaluates various security mechanisms, and provides insights into future research directions to enhance data protection in the cloud.

C. Chang et al. [28] The paper presents an approach for communication between clouds and propose a multi-source information deduplication method based on named entity recognition, improving data consistency and reducing redundancy.

M. Bellare et al. [29] The authors investigated an optimal data encoding parameters based on user preferences for cloud storage, enhancing efficiency and accessibility. Their approach ensures reliable data access without requiring direct communication between clients, making it suitable for distributed environments.

3. PROPOSED SYSTEM

The proposed system is designed to provide a secure and efficient cloud storage solution by integrating block-level deduplication with AES-256 encryption. As cloud adoption increases, managing large volumes of redundant data while ensuring strong security becomes a critical challenge. Traditional deduplication techniques often face conflicts with encryption schemes because encrypted data appears unique even when the underlying plaintext is identical. To overcome this, our system ensures that deduplication and encryption coexist effectively without compromising user confidentiality.

The methodology begins with authenticated user login, ensuring that only verified users are allowed to upload or download files. Once authenticated, users can initiate the file upload process. The system first splits the uploaded file into fixed-size 1KB blocks, providing a fine-grained level of data management that improves deduplication accuracy compared to conventional file-level approaches. Each block is then processed using the SHA-256 hashing algorithm to generate a unique, collision-resistant 256-bit identifier. This identifier acts as a fingerprint for the block, enabling the system to detect redundant blocks with high reliability.

If a block's hash already exists in the system's metadata database, it is marked as a duplicate, and instead of uploading it again, the system updates reference pointers to associate the block with the new file. This significantly saves storage space and reduces cloud bandwidth usage. However, if a block is identified as unique, it is passed through the AES-256 encryption module, where it is encrypted with a dynamically generated secure key. This ensures that even if cloud storage is compromised, the encrypted blocks provide no useful information to unauthorized parties. The encrypted blocks are then uploaded securely to the cloud storage server (such as DriveHQ), ensuring that only protected data resides in the cloud.

In parallel, a relational metadata database is updated to maintain critical information such as block hashes, file-to-block mappings, encryption keys, and user access keys. This metadata is essential for supporting efficient file reconstruction during downloads while maintaining secure access control.

When a user requests to download a file, the system again enforces user authentication and secret key validation. Upon successful verification, it retrieves the associated block hashes and encryption keys from the metadata. The encrypted blocks are downloaded from the cloud, decrypted using the stored AES keys, and sequentially reassembled into the original file format. This process ensures data confidentiality during storage and provides seamless, accurate file reconstruction for authorized users.

The proposed framework offers multiple advantages: it optimizes cloud storage by reducing redundancy at the block level, ensures data confidentiality through robust AES-256 encryption, and maintains the efficiency and speed required for practical deployment. By avoiding unnecessary encryption or uploads for duplicate blocks, the system improves upload and download times, reduces computational overhead, and minimizes cloud storage costs.

Furthermore, the system is designed to be scalable and adaptable to a multi-user environment, supporting scenarios where multiple users may upload similar or identical files. Strong hashing (SHA-256) combined with encryption ensures that even if users share files, deduplication is effective without risking privacy breaches.

Overall, the proposed system successfully bridges the gap between **security** and storage optimization in cloud environments. It lays a strong foundation for future improvements, such as encrypted metadata management, attribute-based access control (ABE), homomorphic encryption for computations on encrypted data, and support for multimedia files in large-scale cloud deployments.

3.1. Methodology:

The methodology adopted in this project involves a secure and efficient process for deduplicating and encrypting user-uploaded files before storing them in a cloud environment. The system employs a step-by-step approach, starting from user authentication to file uploading, splitting, hashing, deduplication, encryption, cloud storage, metadata management, and secure file downloading. Each phase of the process is carefully designed to ensure data integrity, security, and efficient storage utilization.

Process Flow:

- **User Authentication:** The system provides a secure login and registration mechanism to ensure that only authorized users can upload and download files. Upon successful authentication, users are issued secret keys for future file access.
- **File Upload:** The user selects a file through the user interface and initiates the upload process.
- **File Splitting:** The selected file is divided into fixed-size blocks of 1KB. This granularity allows for efficient block-level deduplication and minimizes redundant storage.
- **SHA-256 Hashing:** Each block is hashed using the SHA-256 algorithm to generate a unique 256-bit identifier. This identifier acts as a fingerprint for the block, enabling the system to detect duplicate data blocks effectively.
- **Deduplication Check:** The system queries the database for existing hashes. If a hash match is found, it updates the reference pointer to the existing block, avoiding re-upload. If no match is found, the block is marked for encryption.
- **AES-256 Encryption:** Unique blocks are encrypted using the AES-256 encryption standard. An encryption key is dynamically generated for each block, adding a further layer of security.
- **Cloud Upload:** Encrypted blocks are uploaded to a secure cloud storage platform. The cloud server hosts only encrypted, deduplicated data, ensuring privacy and efficient storage.
- **Metadata Management:** A relational database is used to store metadata, including block hashes, file-block mappings, encryption keys, and user access keys. This information is essential for reconstructing files accurately during download.
- **File Block Mapping and Merging:** File mapping, also known as memory-mapped files, allows a process to access a file's contents as if it were a segment of memory. File merging involves combining two or more files into a single file, which can be done with various tools and techniques depending on the file type and the desired outcome.

- **File Download:** When a user requests a file, the system verifies the access key. It retrieves corresponding metadata, downloads the necessary encrypted blocks, decrypts them, and reconstructs the file in the original sequence before delivering it to the user.

Algorithms Used:

1. SHA-256 Hashing Algorithm

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that takes input data of any size and produces a fixed 256-bit (32-byte) hash value. It is part of the SHA-2 family, developed by the National Security Agency (NSA), and is considered one of the most secure hashing standards in modern cryptography. SHA-256 is deterministic, meaning that the same input will always produce the same hash output. However, even the slightest change in input will drastically alter the output — a property known as the avalanche effect.

This algorithm (Figure 3.1.1) is widely used for ensuring data integrity, verifying digital signatures, authenticating messages, and in applications such as block chain and secure storage systems. In the context of deduplication, SHA-256 serves as a fingerprinting mechanism that can uniquely identify content blocks.

If two blocks produce the same hash, they are assumed to be identical with very high probability. This makes SHA-256 an ideal tool for detecting redundancy in cloud storage systems.

SHA-256 is a member of the SHA-2 family and generates a fixed 256-bit hash value from input data. It is commonly used for data integrity verification, digital signatures, and cryptographic applications.

SHA-256 is commonly used in digital signatures, message authentication, data integrity verification, password hashing, blockchain technology, and secure storage systems. In cloud storage and deduplication contexts, SHA-256 serves as a content fingerprinting mechanism, enabling the identification of redundant data blocks with high accuracy.

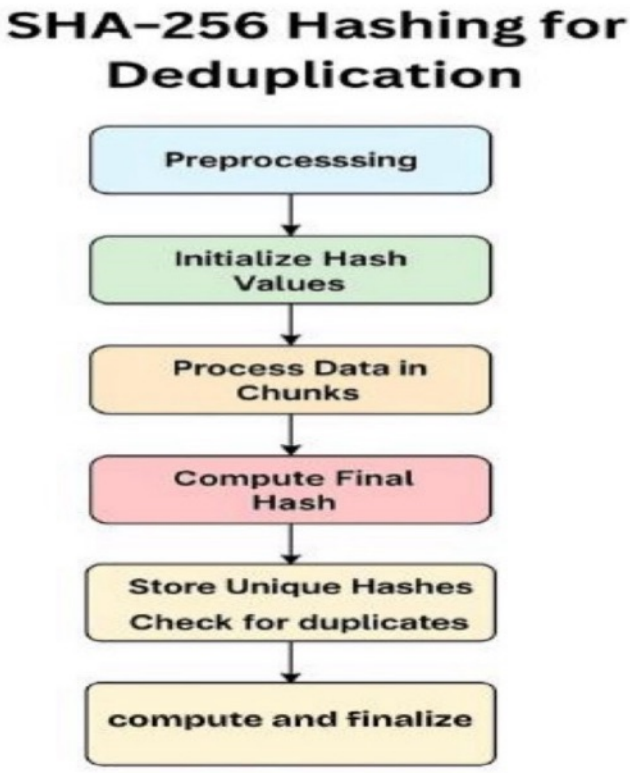


Figure 3.1.1. Flowchart diagram for SHA-256 Hashing Deduplication Algorithm

Steps of the SHA-256 Hashing Algorithm:

Preprocessing: Prepare the input data for hashing by padding it.

Initialize Hash Values: Set the initial hash values for SHA-256.

Process Data in Chunks: Process the input data in 512-bit chunks to compute the hash.

Compute Final Hash: Produce the final 256-bit hash value.

1.Preprocessing

The message is padded(Figure 3.1.1) so that its length is congruent to 448 modulo 512. Add a single '1' bit followed by the necessary number of '0' bits, followed by the original message length as a 64-bit integer.

2. Initialize Hash Values

Set the initial hash values as follows:

H0 = 0x6a09e667

H1 = 0xbb67ae85

H2 = 0x3c6ef372

H3 = 0xa54ff53a

H4 = 0x510e527f

H5 = 0x9b05688c

H6 = 0x1f83d9ab

H7 = 0x5be0cd19

3. Process Data in Chunks

Split the padded message into 512-bit chunks(Figure 3.1.1). For each chunk:

- Create a message schedule of 64 32-bit words.
- Initialize working variables (a, b, c, d, e, f, g, h).
- Perform 64 iterations of the compression function to update the working variables.

4. Compute Final Hash

After processing all chunks, add the working variables to the initial hash values to compute the final digest.

Algorithm:

```
function SHA256(message):
    padded_message = pad_message(message)
    H = [H0, H1, H2, H3, H4, H5, H6, H7]
     for each chunk in padded_message:
        W = create_message_schedule(chunk)
        a, b, c, d, e, f, g, h = H
         for i from 0 to 63:
            T1 = h + Σ1(e) + Ch(e, f, g) + K[i] + W[i]
```

T2 = Σ0(a) + Maj(a, b, c)

a = (T1 + T2) % 2^32

H = [(H[0] + a) % 2^32, (H[1] + b) % 2^32, ..., (H[7] + h) % 2^32]

return concatenate(H)

SHA-256 serves as a highly secure and reliable cryptographic hash function, essential for verifying the integrity and authenticity of data. It transforms any input data into a fixed 256-bit output, ensuring consistency and non-reversibility. Even a minor change in the input leads to a drastically different output, known as the avalanche effect, making it robust against tampering and collisions. The hashing process (Figure 3.1.1) involves preprocessing of the input, chunk-wise compression through logical and arithmetic operations, and final hash computation. Within the scope of this project, SHA-256 is used to generate unique identifiers for 1KB file blocks during deduplication. By comparing these hashes with stored values in the metadata database, the system effectively identifies and filters out redundant blocks, optimizing storage usage and enhancing performance while ensuring data integrity.

Additionally, SHA-256's deterministic nature guarantees that the same input will always yield the same hash, which is crucial for consistent deduplication. The low probability of hash collisions further ensures that distinct data blocks are not mistakenly identified as duplicates. This function is also computationally efficient, making it suitable for real-time or large-scale data processing. Its widespread adoption in standards like blockchain and SSL certificates underlines its trustworthiness and industry relevance. In this project, its integration into the deduplication workflow strengthens both operational efficiency and security posture.

The algorithm's ability to process input in 512-bit blocks allows it to handle large files by breaking them into smaller, manageable units, aligning well with the deduplication process. SHA-256 operates without the need for a secret key, which simplifies its integration into systems where confidentiality is not required but integrity is critical. Furthermore, since SHA-256 is part of the SHA-2 family developed by the NSA and approved by NIST, it meets stringent security standards,

ensuring long-term viability. It is also resistant to known cryptographic attacks, including pre-image and second pre-image attacks, which enhances the reliability of hash-based comparisons in data deduplication systems. Given these properties, SHA-256 is an ideal fit for secure, scalable storage optimization in both enterprise and cloud environments.

2. Deduplication Algorithm Based on SHA-256

To optimize cloud storage by detecting and eliminating duplicate data blocks using their SHA-256 hashes. This ensures that only unique blocks are stored and that duplicate blocks are referenced efficiently, conserving bandwidth and storage space.

The deduplication process leverages the cryptographic strength of SHA-256 to create a unique identifier (hash) for every data block. These identifiers are then used to compare new uploads against existing stored data. If a match is found, the system avoids storing the duplicate block again, updating internal references instead. This approach enables fine-grained storage efficiency at the block level rather than at the file level, which is crucial in environments where small modifications to files are common.

To eliminate redundant data blocks using their SHA-256 hashes and maintain a reference for duplicates.

data_blocks: List of 1KB blocks from a file

hash_database: Set or database of previously seen hash values

unique_blocks: List of blocks not seen before

reference_table: Map of file blocks to existing hash references

Algorithm:

Function Deduplicate_Using_SHA256(data_blocks):

Initialize:

unique_blocks ← []

reference_table ← {}

hash_database ← existing_hashes_from_metadata()

```
For each block in data_blocks:
    hash_value ← SHA256(block)
    If hash_value exists in hash_database:
       reference_table[block] ← hash_value  // Duplicate
    Else:
        unique_blocks.append(block)
       reference_table[block] ← hash_value
Return unique_blocks, reference_table
```

The primary goal of using AES-256 in this project is to ensure confidentiality of user data stored in the cloud. This ensures that even if cloud-stored data is accessed without authorization, the encrypted contents remain unintelligible and unusable without the correct decryption keys.

This algorithm follows the hashing phase and identifies whether a block needs to be stored or simply referenced. If a block's hash already exists, the system avoids re-uploading and only updates references. Otherwise, the block is encrypted and uploaded. This process conserves cloud space and improves efficiency.

AES-256, a symmetric key encryption algorithm, offers a strong level of security by using a 256-bit key, making brute-force attacks computationally infeasible. Its integration with the deduplication process ensures that only unique blocks undergo encryption, which minimizes overhead while maintaining data protection. The encryption process is performed on a per-block basis, allowing fine-grained control and parallel processing for scalability. Keys are securely managed and stored, using either a centralized key management system or per-user derived keys to further enhance security.

Each encrypted block includes metadata for key retrieval and integrity verification, allowing the system to reconstruct files reliably during download. Additionally, combining encryption with deduplication balances confidentiality and storage efficiency—a challenge in many secure storage systems. AES-256's widespread support across platforms and hardware acceleration options also contribute to its suitability for high-performance applications. Its role in this system is essential to

uphold user trust and comply with data protection standards, especially in multi-tenant or public cloud environments.

3. AES-256 Encryption Algorithm

AES-256 (Advanced Encryption Standard - 256 bit) is a symmetric key encryption algorithm that is part of the widely adopted AES family, standardized by the U.S. National Institute of Standards and Technology (NIST). As a symmetric cipher, it uses the same key for both encryption and decryption, which ensures simplicity and speed in secure data communication. It operates on 128-bit blocks of data and performs 14 rounds of transformation processes including substitution, permutation, mixing, and key addition.

These operations introduce confusion and diffusion, making it extremely resistant to cryptanalytic attacks. Due to its high level of security and performance, AES-256 is widely used in applications such as VPNs, secure file storage, messaging, and full disk encryption.

Its 256-bit key length offers a vastly larger key space compared to AES-128 or AES-192, significantly increasing resistance against brute-force attacks. Moreover, hardware acceleration support through modern CPUs (e.g., Intel AES-NI) allows AES-256 to deliver strong encryption without compromising processing speed. The algorithm is also known for its predictability and reliability, making it ideal for systems that require both security and stability under high workloads. In cloud-based storage environments, AES-256 helps meet compliance requirements such as GDPR, HIPAA, and ISO/IEC 27001.

When combined with secure key management practices, it provides a robust foundation for data confidentiality and access control. Its deterministic nature, when used in appropriate modes like CBC or GCM, allows for consistent encryption outcomes while maintaining data integrity. AES-256's balance of strong security and computational efficiency makes it a critical component in building secure, scalable, and trusted systems.

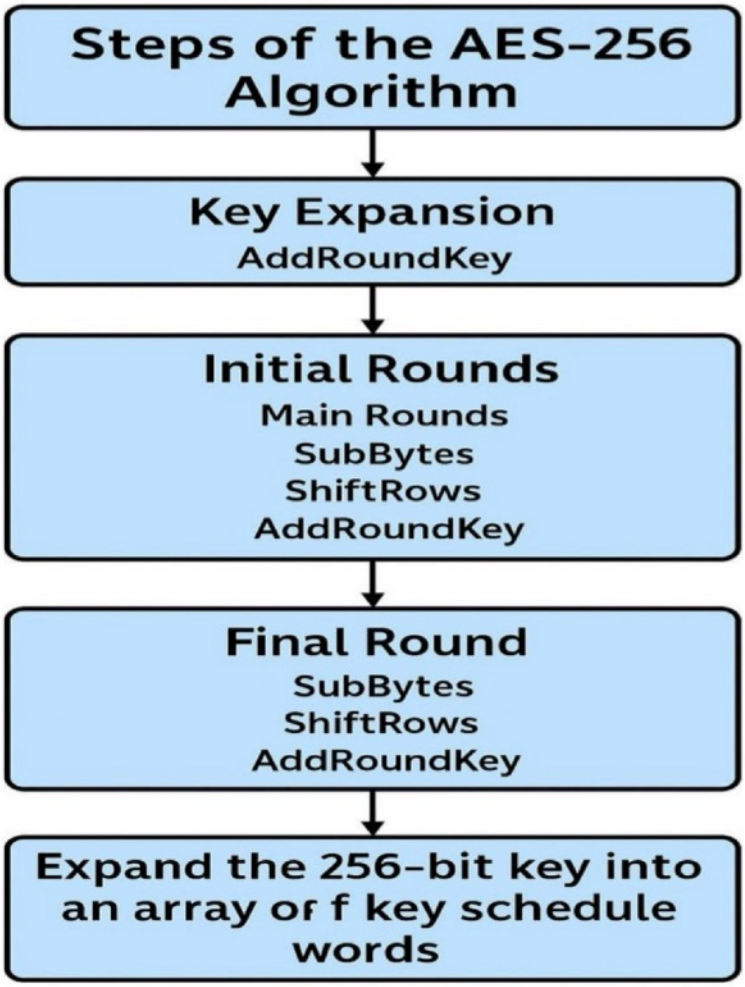


Figure 3.1.2: Flowchart diagram for the AES-256 Algorithm

Encrypted block ready for cloud upload. A 1KB data block determined to be unique indicates that the system has verified this block does not duplicate any previously stored or uploaded block, likely using a hashing algorithm for comparison. This process is typical in data deduplication systems, which aim to optimize storage by avoiding redundant uploads.

Steps of the AES-256 Algorithm

1. Key Expansion:

The original key is expanded into a total of 60 words (4 bytes each) for AES-256(Figure 3.1.2), resulting in a total key schedule of 240 bytes. Expand the 256-bit key into an array of key schedule words, likely using a hashing algorithm for comparison.

2. Initial Round:

AddRoundKey: Combine the plaintext with the first round key using bitwise XOR.

Each byte of the block is combined with the corresponding round key using bitwise XOR.

3. Main Rounds: Repeat for 13 rounds:

SubBytes: Substitute bytes using a fixed substitution box (S-box).
ShiftRows: Circularly shift rows of the state array. Row 0 is not shifted, Row 1 is shifted left by 1, Row 2 is shifted left by 2, and Row 3 is shifted left by 3.

MixColumns: Mix the columns to provide diffusion. Each column of the state is mixed independently, providing additional diffusion. The current block is XORed with the round key for this round.

4. Final Round:

SubBytes: Substitute bytes.
ShiftRows: Shift rows.

AddRoundKey: Combine with the last round key.AES operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, or 256 bits. The algorithm transforms plaintext into ciphertext through multiple rounds of substitution, permutation, and mixing operations.

Algorithm:

```
function AES_Encrypt(Plaintext, Key):
    KeySchedule = KeyExpansion(Key)
    State = AddRoundKey(Plaintext, KeySchedule[0])
    for round from 1 to 13:
        State = SubBytes(State)
        State = ShiftRows(State)
        State = MixColumns(State)
        State = AddRoundKey(State, KeySchedule[round])
    State = SubBytes(State)
```

State = ShiftRows(State)

State = AddRoundKey(State, KeySchedule[14])

return State

The integration of SHA-256 and AES-256 algorithms (Figure 3.1.2) formed the backbone of the proposed secure cloud storage system, delivering an effective solution that balanced both data security and storage efficiency. Through the intelligent application of SHA-256 hashing, the system was able to accurately identify duplicate data blocks at a fine-grained level (1KB), drastically reducing the amount of redundant data uploaded and stored in the cloud. This not only minimized storage space but also resulted in noticeable improvements in upload and download times, as fewer blocks needed to be encrypted, transmitted, and stored.

SHA-256 proved to be particularly well-suited for this task due to its strong collision resistance, and speed. Even small changes in the input data led to entirely different hash outputs, ensuring high accuracy in redundancy detection. By comparing block hashes instead of raw content, the system avoided the need for computationally expensive block comparisons, thereby maintaining excellent performance even as the dataset size grew.

On the other hand, AES-256 encryption was seamlessly integrated into the framework to ensure that confidentiality was never compromised, even as storage efficiency was pursued. AES-256, with its 256-bit key length, provided a very high level of security against brute-force attacks and unauthorized access. Each unique block was encrypted independently with a dynamically generated AES-256 key, ensuring that even if the encrypted cloud data were intercepted or accessed, it would remain undecipherable without the corresponding key. The encryption and decryption operations were computationally efficient and did not introduce significant delays, confirming that strong security could be achieved without sacrificing system responsiveness.

The combination of SHA-256 and AES-256 allowed the system to operate with a layered defense strategy — deduplication at the block level to optimize storage, and encryption at the block level to ensure privacy. The metadata management further strengthened the architecture by securely associating block hashes, encryption keys, and file mappings without exposing sensitive information.

Advantages of this methodology:

- Reduces redundant data storage: By using SHA-256 hashing to generate a unique fingerprint for each data block, systems can detect and eliminate duplicate content. When multiple identical blocks are identified, only one copy is stored, and references are maintained to it, significantly reducing storage requirements.
- Ensures data confidentiality and integrity: SHA-256 helps verify that data has not been tampered with during transmission or storage. When combined with encryption techniques, it also ensures that unauthorized users cannot access or alter sensitive information, maintaining both its confidentiality and authenticity.
- Enhances storage optimization and retrieval efficiency: Since duplicate data blocks are stored only once, less physical storage space is used. This optimization speeds up data retrieval processes, as the system handles fewer blocks and can access the required data more quickly.
- Facilitates scalable and secure cloud storage management: In cloud environments, where large volumes of data are handled continuously, SHA-256-based deduplication allows the system to manage growing data loads efficiently while maintaining high security standards. This scalability is critical for accommodating increasing user demands.
- Mitigates risks associated with data breaches by encrypting each unique block individually: By treating each deduplicated block as a separate unit and encrypting it independently, the system ensures that even if one block is compromised, others remain secure. This granular encryption approach limits the impact of potential data breaches.

3.1.1 Architecture Diagram

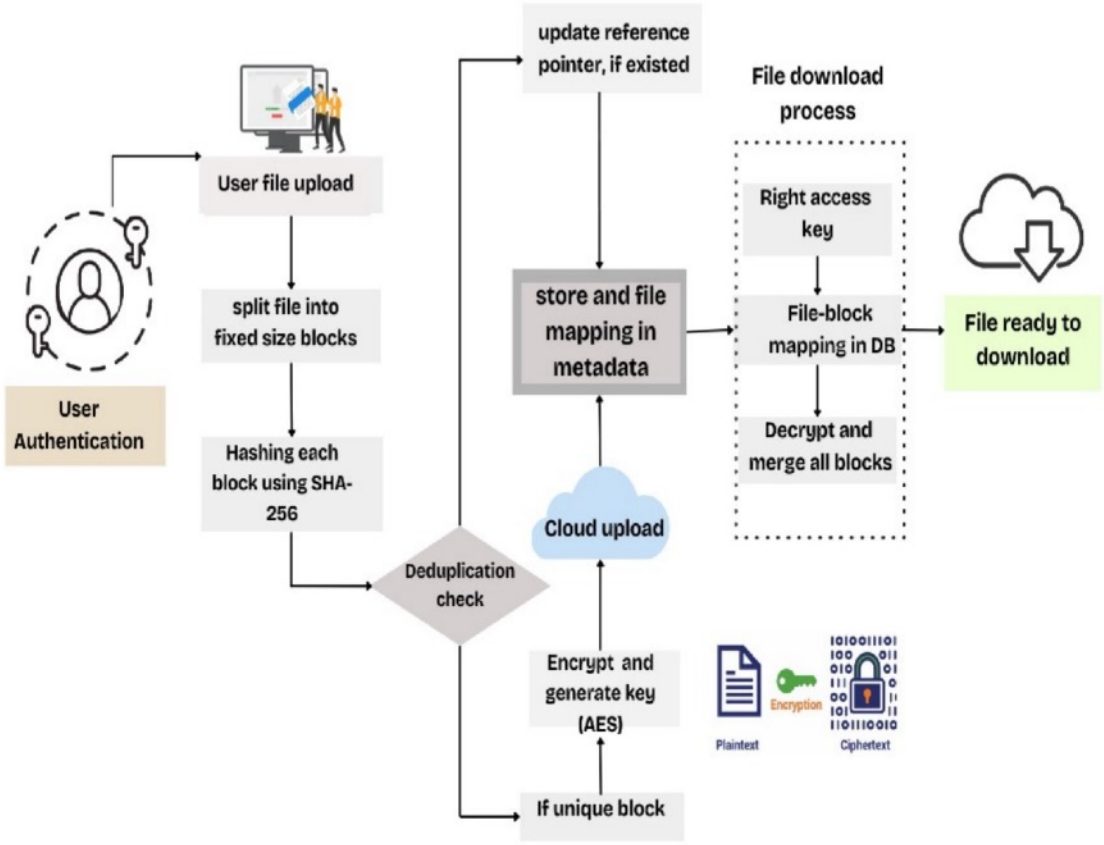


Figure 3.1: Architecture of Secure Cloud Storage with Block-Level Deduplication and AES-256 Encryption

The architecture of the proposed secure deduplication system is carefully designed to balance performance, storage efficiency, and data security. It systematically separates responsibilities into interconnected components that work together to ensure seamless file management, from upload to secure storage and later retrieval. The process begins with the user interface, where the authenticated user uploads a file. The file undergoes file splitting, dividing it into equal-sized 1KB blocks. This enables block-level deduplication — a finer and more efficient alternative to traditional file-level deduplication (Figure 3.1.)

Each of these blocks is processed by the Hashing Module, where a SHA-256 hash is computed. These hashes act as unique digital identifiers for the data blocks. Once the hash is generated, the Deduplication Module consults the metadata database to check whether a block with the same hash already exists. If it does, the system

updates a reference pointer to the existing block, avoiding redundant storage. For blocks that are identified as unique, the Encryption Module encrypts them using AES-256 encryption (Figure 3.1.). The encryption process includes generating dynamic keys to enhance data security further, ensuring each block's confidentiality before cloud storage. The Cloud Storage Module then securely uploads these encrypted blocks to a remote cloud storage platform (DriveHQ or a similar secure FTP-based cloud). This ensures that only encrypted, deduplicated content is stored, minimizing storage use and improving security.

Meanwhile, the Metadata Management Module updates the relational database with information about the uploaded block hashes, the mapping of blocks to their files, and their encryption keys. This metadata is crucial for reconstructing the original files accurately. During the file download process, the user must input the correct access key, which is verified by the User Authentication Module. Upon successful verification, the system queries the metadata database to retrieve the mappings of the file's blocks and their corresponding encryption keys.

The File Download and Reconstruction Module downloads the encrypted blocks from the cloud, decrypts them using AES-256 keys, and reassembles the file by ordering the blocks in sequence according to the metadata. Finally, the system presents the reconstructed file to the authenticated user for download. This end-to-end workflow ensures that data remains secure, redundant storage is eliminated, and system performance is maintained at scale. The modular design also supports future enhancements such as key rotation, multi-user access control, and integration with zero-knowledge proof mechanisms for even stronger privacy guarantees. Overall, the architecture provides a comprehensive and adaptable solution for secure, efficient cloud storage.

This architecture ensures:

- High storage optimization by preventing duplicate storage at a block level.
- Data confidentiality and integrity through AES-256 encryption and SHA-256 hashing.
- Seamless file management and retrieval using well-maintained metadata and access control.

3.1.2 Functional Modules

The proposed system comprises several interrelated functional modules, each responsible for handling specific tasks in the secure deduplication process.

3.1.2.1. User Authentication Module: This module manages the entire user authentication process, which includes user registration, login, and credential verification. Once authenticated, users are granted secure access keys, enabling them to perform upload and download operations within the system. The module also records user activities and maintains detailed logs to enhance system security and accountability.

3.1.2.2. File Splitting Module: The file splitting module handles the preprocessing of uploaded files. It divides each file into uniform 1KB blocks, facilitating efficient deduplication by enabling block-level comparisons (Figure 3.1.1.1). Accurate block mapping is maintained to ensure correct file reconstruction during download.

3.1.2.3. Hashing Module: Responsible for generating unique hash values for each data block, this module uses the SHA-256 algorithm to produce a 256-bit identifier per block. The generated hashes act as digital fingerprints, allowing the system to detect duplicate blocks reliably and guarantee data integrity by avoiding hash collisions.

3.1.2.4. Deduplication Module: This module performs the deduplication check by comparing the newly generated block hashes against the existing hash values stored in the metadata database. If a match is detected, indicating a duplicate block, it updates the reference pointer, avoiding redundant data storage. Unique blocks are forwarded to the encryption module.

The deduplication process can be formally expressed as follows:

File Splitting: Let a file F be divided into fixed-size blocks:

$$F = \{B1, B2, B3, ..., Bn\}$$

Hash Calculation: Each block Bi is hashed using SHA-256 to generate a unique identifier:

$$H(Bi) = SHA\text{-}256(Bi)$$

Duplicate Detection: The system checks whether H(Bi) exists in the deduplication index:

If $H(Bi) \in D$, then reference existing block, else store new.

Storage Optimization: If H(Bi) exists, a reference is created instead of storing the block. If it does not exist, Bi is encrypted using AES encryption and stored:

$$E(Bi) = AES(Bi,K)$$

Reference Linking: For each duplicate block, a reference link is maintained in the metadata table:

$$R(Bi) = \text{Pointer to existing block in storage}$$

Integrity Verification: To maintain data integrity, stored blocks undergo periodic hash validation.

$$\text{Check } H(Bi) \text{ ? } = SHA256(Bi)$$

By implementing block-level deduplication and AES encryption, the system achieves efficient storage management while ensuring high data security. Access is restricted through strict access control mechanisms, ensuring that only authorized users can retrieve specific data blocks.

3.1.2.5. Encryption Module: Dedicated to securing unique data blocks, the encryption module applies AES-256 encryption to each block. It dynamically generates unique encryption keys for each block, encrypts the data independently, and ensures that deduplication efficiency is preserved while protecting against unauthorized access and inference attacks.

The module operates in parallel with the deduplication engine to minimize latency during upload. It also supports secure key storage and key regeneration policies for enhanced long-term data confidentiality.

To improve performance, the module may utilize hardware acceleration (e.g., AES-NI) when available. Encryption metadata, such as initialization vectors and key identifiers, is securely stored to support reliable decryption during file retrieval.

The design ensures that no plaintext data is ever stored or transmitted, adhering to a zero-trust architecture for data handling.

3.1.2.6. Cloud Storage Module: This module handles the secure upload of encrypted blocks to the cloud server (such as DriveHQ). It ensures that only deduplicated, encrypted data occupies cloud storage space, enhancing both security and storage efficiency. The module also supports a scalable, distributed storage infrastructure. It includes mechanisms for fault tolerance, data replication, and encrypted communication channels (e.g., SFTP/FTPS) to safeguard data in transit and at rest. The module can interface with multiple cloud platforms, providing redundancy and availability through multi-cloud deployment strategies. It also handles metadata synchronization and storage zone management to optimize performance across geographic locations.

3.1.2.7. Metadata Management Module: The metadata management module is responsible for storing and managing block hashes, file-block mappings, encryption keys, and user access keys within a relational database. It facilitates accurate file reconstruction, efficient storage retrieval, and secure key management, while maintaining comprehensive audit logs for system monitoring and security tracking.

3.1.2.8. File Download and Reconstruction Module: When a user requests a file, this module first verifies the user's access key. It then retrieves the relevant block metadata, encrypted blocks, and decryption keys from the database and cloud storage. The module decrypts individual blocks using AES-256 and reconstructs the original file by merging the decrypted blocks in the correct sequence. Finally, the complete file is made available for the user to securely download.

Module Interactions: These functional modules interact through structured data flows and metadata exchanges. The Metadata Management Module coordinates deduplication checks, file-block mappings, and encryption key retrievals. Authentication services control access permissions for initiating upload and download operations. Decryption and file reconstruction processes are tightly integrated with access validation, ensuring seamless, secure interactions between modules. Collectively, these modules create a reliable, efficient, and secure cloud-based deduplication and storage system optimized for data protection and space

management. The design ensures modular scalability, allowing individual modules to be upgraded or replaced independently without disrupting the entire framework. For instance, newer encryption standards or alternative hashing methods can be incorporated with minimal code restructuring. Inter-module communication is secured and logged to ensure full traceability and prevent unauthorized operations. This architecture also supports concurrent multi-user access, with each user's activity securely isolated and tracked.

4. EXPERIMENTAL SETUP & RESULTS

4.1. System Specifications:

The system was developed and tested in a controlled environment to simulate secure file storage, block-level deduplication, and encrypted cloud uploads. The specifications ensure that the system operates efficiently with minimal computational overhead while maintaining high security standards. Below are the detailed hardware and software requirements used for the development, deployment, and evaluation of the proposed secure deduplication framework.

4.1.1 Software Requirements:

Database Server	: MySQL
Database Client	: SQL yog
Server	: Apache Tomcat
Platform	: Java
Technology	: Servlets, JSP, JDBC
Client Side Technologies	: Html, CSS, Java Script
IDE	: Eclipse
Testing	: Junit

Technical Specifications and Descriptions

Software Requirements

Database Server: MySQL

MySQL is an open-source relational database management system (RDBMS) widely used for efficient data storage, retrieval, and management. It ensures the integrity of stored data and supports SQL-based querying, making it a reliable choice for cloud-based applications.

Database Client: SQL Yog

SQL Yog is a graphical client used for managing MySQL databases. It provides an intuitive interface to interact with the database, execute queries, manage tables, and

optimize database performance. It simplifies database administration and enhances productivity.

Server: Apache Tomcat

Apache Tomcat is an open-source Java-based web server and servlet container used for deploying and running Java applications. It processes JSP (Java Server Pages) and Servlets, making it ideal for web-based cloud applications. It offers high scalability, security, and efficient request handling.

Platform: Java

Java is the primary programming language used in the project. It is platform-independent, object-oriented, and highly secure, making it an ideal choice for building enterprise-level applications, including secure cloud storage systems.

Technology: Servlets, JSP, JDBC

- Servlets**:** Java Servlets serve as the cornerstone of the back-end architecture in the proposed system. These server-side Java programs handle HTTP requests and responses, functioning as the web application's controller layer. Servlets manage critical aspects such as business logic, session tracking, and communication between the front-end (JSP) and back-end (JDBC/MySQL). They are responsible for orchestrating the entire file upload and processing workflow—receiving files from users, splitting them into smaller 1KB blocks, calculating hash values using SHA-256, and ensuring that deduplication is applied efficiently. Additionally, Servlets are tasked with handling the encryption and decryption of these blocks using AES-256 encryption, ensuring data confidentiality during storage and transfer. They also manage user authentication, directing requests to relevant operations such as file retrieval, upload, and download. The multithreading capabilities of Servlets enable the application to efficiently handle concurrent user requests, ensuring scalability and responsiveness even under heavy loads. By maintaining persistent connections and managing state, Servlets play a critical role in ensuring that the application performs smoothly and reliably.

- JSP (Java Server Pages): JavaServer Pages (JSP) serve as the presentation layer of the application, providing a dynamic and interactive interface for users. JSP allows Java code to be embedded directly within HTML pages, enabling the server to generate dynamic content based on user interactions. In the context of the proposed system, JSP is responsible for displaying real-time information to users, such as file upload progress, available file listings, and any error messages generated during the file processing or retrieval stages. The separation of presentation and business logic between JSP and Servlets simplifies the development and maintenance of the system, as changes to the user interface do not directly affect the back-end logic. This separation also facilitates easier updates and improvements to the user experience without disrupting core functionalities. Furthermore, JSP enables quick and easy development of interactive and responsive web pages, contributing to a smooth and intuitive user experience while interacting with the underlying services provided by the Servlets.

- JDBC (Java Database Connectivity): Java Database Connectivity (JDBC) is a crucial component that allows the system to interact with the relational database (e.g., MySQL) to store and retrieve metadata related to the file blocks, hashes, encryption keys, and user credentials. JDBC acts as the intermediary between the Servlets and the database, enabling secure, efficient database operations through the execution of SQL queries and updates. It is essential for managing the mapping between uploaded file blocks and their corresponding encryption keys, ensuring accurate and efficient file reconstruction during download. JDBC supports prepared statements, which not only enhance performance by reducing the overhead of SQL query preparation but also help prevent SQL injection attacks, ensuring the security and integrity of the system. Moreover, JDBC's transactional capabilities ensure that operations involving the metadata database are consistent and reliable, particularly during operations like file uploads, deletions, and reconstructions. The seamless integration of JDBC with the other components guarantees smooth and real-time data interaction, ensuring that file processing workflows are executed correctly and that system performance is maintained across a growing volume of data.

Client-Side Technologies: HTML, CSS, JavaScript:

These technologies are essential for creating the user-facing part of a web application.

- **HTML:**
  HTML serves as the backbone of web development. It provides the basic structure and content of a webpage, such as headings, paragraphs, links, images, tables, and forms. HTML uses a system of tags and attributes to define elements and their roles within the document.
- **CSS:**
  CSS is used to control the presentation and layout of HTML elements. It allows developers to apply styles like colours, fonts, spacing, and positioning, making the user interface visually appealing and consistent. CSS also supports responsive design techniques.
- **JavaScript:**
  JavaScript is a powerful client-side scripting language that enables interactive features and dynamic content on web pages. It can respond to user actions, manipulate the DOM, communicate with back-end services via APIs (e.g., using fetch or AJAX), and handle asynchronous operations.

4.1.2 Hardware Requirements:

Processor	: I3
Ram	: 4GB.
Hard Disk	: 500 GB.

Processor: Intel Core i3

The system requires at least an Intel Core i3 processor to handle computations, encryption, and database operations efficiently. It ensures smooth execution of deduplication processes and secure file handling.

RAM: 4GB

A minimum of 4GB RAM is required to manage simultaneous processes, including encryption, database queries, and file deduplication, without affecting system performance.

Hard Disk: 500GB

The storage capacity should be at least 500GB to accommodate database records, encrypted files, deduplicated blocks, and metadata. It ensures sufficient space for handling cloud storage requirements.

The chosen technical stack ensures a secure, scalable, and efficient cloud storage system. MySQL and SQLyog facilitate structured data storage and management, while Java, Servlets, JSP, and JDBC enable a dynamic and interactive user experience. Apache Tomcat ensures seamless web server operations, while Eclipse provides an efficient development environment. The integration of UML and testing tools enhances system design and reliability. The hardware requirements support optimal performance, ensuring secure deduplication without excessive computational overhead.

Administrative user interface

The 'administrative user interface' concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. These interfaces help the administrators with all the transactional states like Data insertion, Data deletion and Date updating along with the extensive data search capabilities.

It enables administrators to securely control and maintain the core transactional data of the system.

The operational or generic user interface:

It helps the end users of the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information in a customized manner as per the included Flexibilities. These interfaces are designed to be intuitive and easy to use, allowing users to navigate and perform tasks without needing extensive technical knowledge. Effective user interfaces are designed to be accessible to a wide range of users, including those with disabilities. It empowers users to navigate and use system features effortlessly, with personalization and accessibility in mind.

4.2 Description:

Upload process:

This project was developed and tested on a system meeting the specifications mentioned.. The proposed system integrates various technologies and tools for implementing secure cloud storage using deduplication and encryption. The development environment included Java for the backend logic, JSP and Servlets for the web interface, MySQL for managing metadata, and DriveHQ for cloud storage.

Figure 4.2.1 User Login and new User Registration page.

The login page is the primary access point for both administrators and regular users, providing secure entry into the system. It features input fields where users must enter their registered username and password to authenticate themselves before gaining access to the application.

Behind the interface (Figure 4.2.1), authentication logic validates the credentials against the stored database records, ensuring that only authorized users can interact with the system's secure storage and deduplication functionalities.

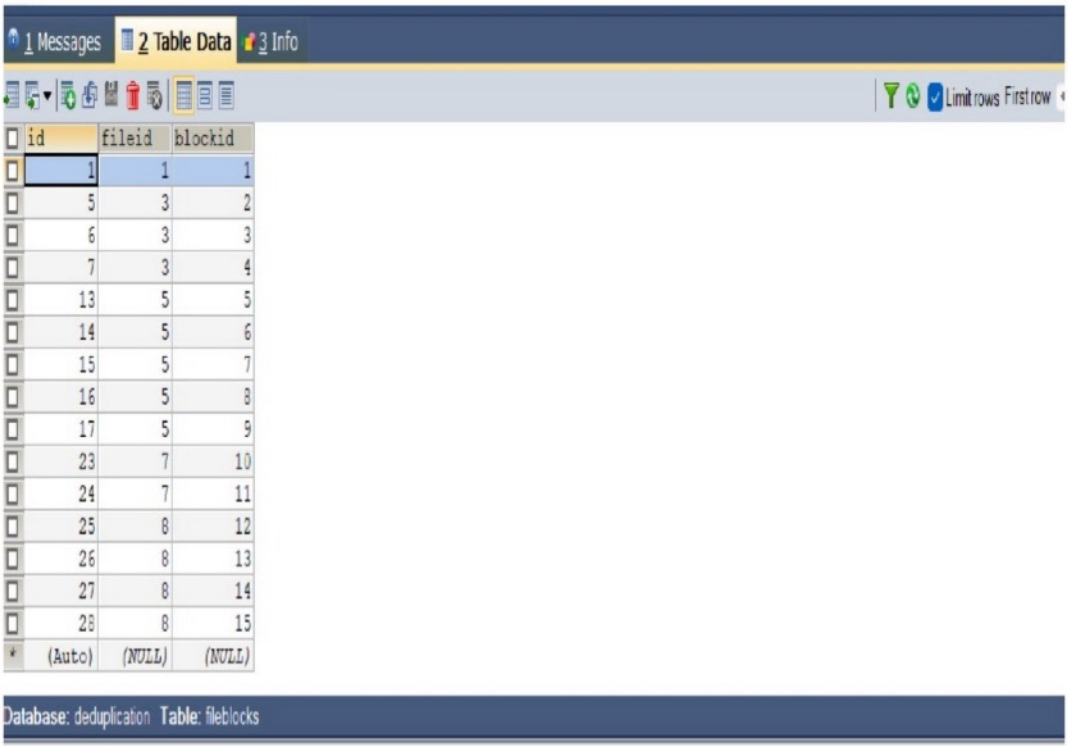
Figure 4.2.2. User Performing Action - Uploading File into the cloud

Users were able to log in (Figure 4.2.2), upload files, and securely store data on the cloud. Files were split into 1KB blocks, hashed using SHA-256, checked for duplication, encrypted using AES-256, and uploaded to the cloud server. The process where a user can upload files to a server or cloud storage service begins with user authentication, ensuring that only authorized individuals can access and manage their data. The system verifies credentials via a login interface, where users input their username and password. Upon successful authentication, users are redirected to a personalized dashboard where they can easily manage their uploaded files, view file statuses, and track progress.

Once a user selects a file for upload, the system initiates the file processing phase. The file is split into smaller 1KB blocks, which ensures that even large files can be processed efficiently. Each block is hashed using SHA-256, generating a unique fingerprint to check for duplication across the cloud storage. If the block has been previously uploaded, the system references the existing block, avoiding redundant storage. If the block is new, it is encrypted using AES-256 with a unique key for enhanced security before being uploaded to the cloud. This encryption ensures that

data remains confidential and protected from unauthorized access, even if intercepted during transmission.

The cloud storage system not only ensures secure file storage but also enables quick retrieval and efficient deduplication, allowing users to save storage space and maintain high performance. The user-friendly dashboard also provides real-time feedback on the upload process, ensuring that users are informed and confident throughout their interactions with the system. This streamlined process enhances both security and usability, making cloud storage more accessible and efficient for users.



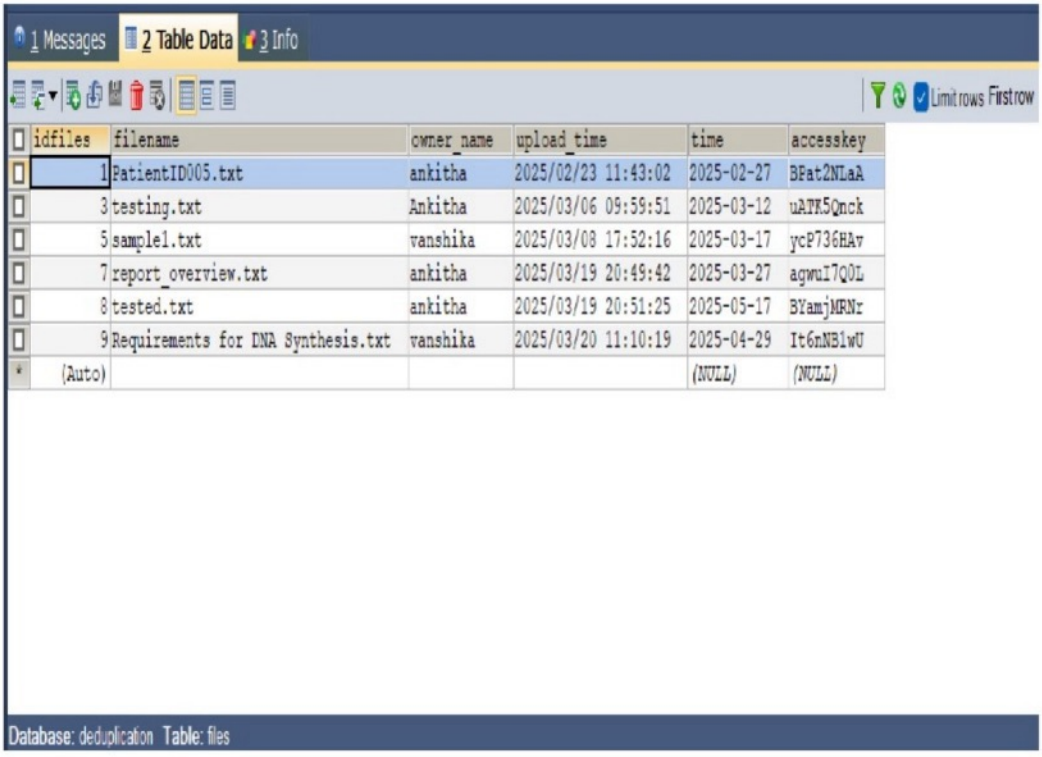
id	fileid	blockid
1	1	1
5	3	2
6	3	3
7	3	4
13	5	5
14	5	6
15	5	7
16	5	8
17	5	9
23	7	10
24	7	11
25	8	12
26	8	13
27	8	14
28	8	15
(Auto)	(NULL)	(NULL)

Figure 4.2.3. File- Block mapping in Database Server

Each uploaded file is assigned a unique file ID and stored in the database along with metadata such as the filename, uploader's name, upload timestamp, and a randomly generated AES-based access key. This access key is securely stored and used later for decrypting the file during retrieval, ensuring that only the authorized user can access the content. The uploaded file is split into smaller blocks, and each block is identified with a unique block ID, facilitating a more granular approach to file management. The mapping between file IDs and block IDs is maintained in a separate table (Figure 4.2.3), which allows the system to reference and reuse

existing blocks when identical data is uploaded by another user, significantly improving storage efficiency. This block-to-file mapping ensures efficient deduplication, reduces redundant storage, and enables secure reconstruction of the original file using the corresponding access key, even if multiple users upload the same data.

When a file is requested for download, the system can quickly retrieve the relevant blocks from the cloud and reconstruct the original file with minimal overhead, speeding up the retrieval process and reducing latency. Furthermore, the separation of block-level data mapping from file-level metadata improves data management and retrieval performance by simplifying the database structure and allowing for more efficient indexing. This approach not only optimizes storage and retrieval times but also facilitates better scalability, making it easier to manage and query large amounts of data across users and devices.



idfiles	filename	owner_name	upload_time	time	accesskey
1	PatientID005.txt	ankitha	2025/02/23 11:43:02	2025-02-27	BPat2NLaA
3	testing.txt	Ankitha	2025/03/06 09:59:51	2025-03-12	uATK5Qnck
5	sample1.txt	vanshika	2025/03/08 17:52:16	2025-03-17	ycP736HAv
7	report_overview.txt	ankitha	2025/03/19 20:49:42	2025-03-27	agwuI7Q0L
8	tested.txt	ankitha	2025/03/19 20:51:25	2025-05-17	BYamjMRNr
9	Requirements for DNA Synthesis.txt	vanshika	2025/03/20 11:10:19	2025-04-29	It6nNB1wU
(Auto)				(NULL)	(NULL)

Figure 4.2.4. Secret Key Generation in Database for accessing

The secret key is generated securely and encrypted using a public key before being sent via email. The email contains an encrypted version of the secret key, which can only be decrypted by the recipient using their private key. Upon receiving the email (Figure 4.2.4), the recipient uses their private key to decrypt the secret key and

access the necessary data. This ensures that the key remains secure during transmission, as only the intended recipient has the means to unlock it. The encrypted secret key is stored temporarily and expires after a set period to prevent unauthorized access in case the email is not accessed in time. This time-sensitive nature ensures that any potential vulnerabilities related to stale or unused keys are mitigated. To further safeguard the system, a secure communication channel, such as SSL/TLS, is established for any subsequent key exchanges, ensuring that all further transmissions of sensitive data are protected against eavesdropping or tampering.

Key rotation is implemented periodically to ensure ongoing security. By periodically changing the encryption keys, the system reduces the risk of a compromised key being used over an extended period. The rotation process involves generating new keys, securely distributing them to the recipients, and ensuring that old keys are securely retired and no longer used.

This dynamic approach to key management significantly strengthens the overall security posture of the system. This mechanism ensures that even if the email communication is intercepted, unauthorized users cannot gain access to the file decryption keys, as they would still need the private key of the recipient and a valid, unexpired key to successfully decrypt the data. The combination of encryption, key expiration, secure channels, and key rotation guarantees that data remains protected throughout its lifecycle, from generation to transmission and access.

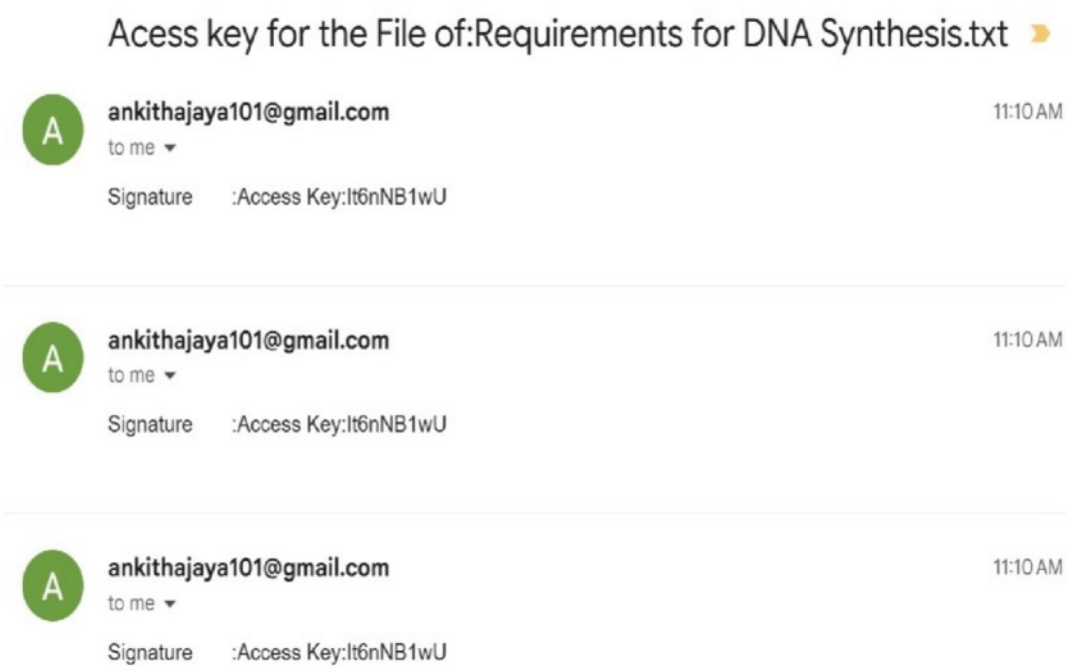


Figure 4.2.5. Secret Key sent through a mail

This (Figure 4.2.5) illustrates the secure transmission of the secret access key to the authorized user via email. After a user successfully uploads a file, the system generates a unique secret key associated with the encrypted file blocks. To maintain confidentiality and secure access, this key is encrypted using a public key encryption method and then sent to the user's registered email address. Only the intended recipient, possessing the corresponding private key, can decrypt and retrieve the secret key securely. This mechanism ensures that even if the email communication is intercepted, unauthorized users cannot gain access to the file decryption keys. The process strengthens the overall security of the system by combining secure key generation, email-based transmission, and asymmetric cryptography, ensuring that file retrieval remains strictly restricted to authenticated and verified users. Additionally, this approach eliminates the risk of key exposure during the file upload process, as the key is never stored or transmitted in plaintext. By leveraging asymmetric encryption, the system adds an additional layer of protection against potential attacks, making it nearly impossible for unauthorized parties to gain access to sensitive decryption information even if they have access to the encrypted file blocks.

Download process:

Figure 4.2.6. User Performing Action - Downloading File from the cloud can be Inter User or Intra User

During the download process (Figure 4.2.6), the system authenticated user requests, validated secret keys, retrieved encrypted blocks from the cloud, decrypted them, and reconstructed the original files.

The user interface of the proposed secure cloud storage system is designed to provide an intuitive, secure, and seamless experience from the moment a user logs in to the final stage of file retrieval. Built using JSP and Servlets on the frontend and supported by backend Java logic, the UI facilitates both administrative and operational functionalities, including file upload, deduplication, encryption, metadata management, secret key generation, and secure download.

Starting from user authentication, the system verifies credentials via a login interface and redirects users to a personalized dashboard. Once logged in, users can upload files through a simplified file selection panel. The UI interacts with backend servlets, which manage splitting the uploaded file into 1KB blocks, generating SHA-256 hashes, and checking the database for duplicate entries. If a block is unique, it is encrypted using AES-256 and then uploaded to the cloud (DriveHQ). Duplicate blocks are skipped and referenced, optimizing storage. During this

process, the UI provides real-time feedback such as upload progress, deduplication status, and encryption confirmation.

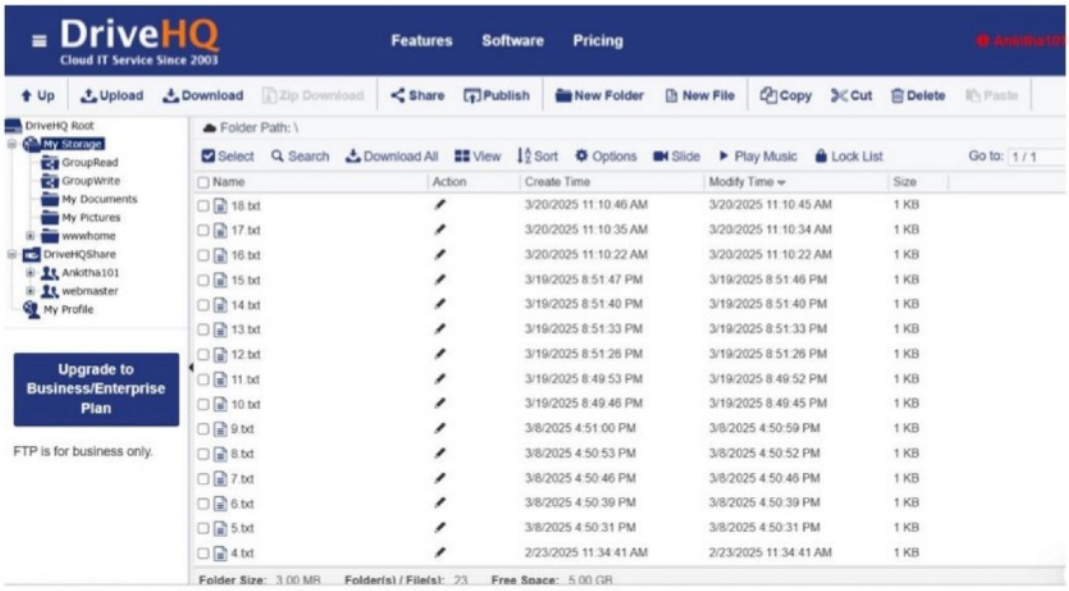


Figure 4.2.7. Drive HQ Cloud interface , where all files are stored.

Overall, the UI (Figure 4.2.7) not only enhances usability but also plays a critical role in enforcing security. It bridges complex backend operations like cryptographic key management and block-level deduplication with a clean, user-friendly experience, ensuring that users can interact with the system effortlessly while remaining informed about their data's status. The UI is designed to provide real-time feedback on file uploads, display clear error messages, and show progress indicators, offering users transparency throughout the process. Features like secure session handling ensure that users' credentials are protected, while access to sensitive information is restricted through proper authentication. Detailed upload/download logs give users an overview of their activities, enabling them to track file operations for accountability and troubleshooting. Furthermore, automatic alerts for key expiry, system errors, or security issues are incorporated to proactively notify users, ensuring timely responses to potential vulnerabilities. The interface also offers easy access to file metadata, encryption status, and decryption key management, ensuring that users can securely retrieve their data without difficulty.

4.3 Results & Test Analysis:

To evaluate the effectiveness of the proposed secure deduplication system, a series of experiments were conducted using 50 text files ranging between 10KB to 20KB in size, hosted on DriveHQ Cloud Storage (with a 5GB free-tier allocation). The analysis compared system performance with and without deduplication across multiple metrics, focusing on storage savings, performance improvements, and cryptographic efficiency.

4.3.1: Performance analysis of de-duplication and encryption:

Table 4.3.1: Performance analysis of de-duplication and encryption for 50 files.

Metric	Without Dedup.	With Dedup.	Improvement
Cloud Stroage Usage	800KB	440KB	45% less
Upload Latency (Avg.)	7.9s	4.8s	39% faster
Retrieval Time (Avg.)	30.6s	21.7s	29% faster
SHA-256 Hash Time	1.5s	1.5s	No Change
AES-256 Encrypt Time	3.9s	3.9s	No Change

Storage Optimization:

The results show that cloud storage usage dropped from 800KB (without deduplication) to 440KB (with deduplication), resulting in a 45% reduction in storage space. This clearly validates the impact of block-level deduplication in minimizing redundant data storage, especially when dealing with multiple similar files across users.

Upload and Retrieval Efficiency:

The average upload latency decreased from 7.9 seconds to 4.8 seconds—a 39% improvement. Similarly, file retrieval time improved by 29%, dropping from 30.6

seconds to 21.7 seconds (Table 4.3.1) . These improvements are largely due to the reduced data volume being transferred and stored, highlighting the practical benefits of deduplication not just in storage but also in transmission speed.

Cryptographic Overhead:

The time taken for SHA-256 hashing and AES-256 encryption remained constant with or without deduplication:

SHA-256 Hash Time: 1.5s

AES-256 Encrypt Time: 3.9s

AES-256 emerged as the most suitable for the project due to its strong security and acceptable performance. As observed in experimental results, AES-256 exhibited an average encryption time of just 2 milliseconds per kilobyte (ms/KB), striking a strong balance between cryptographic strength and speed.

This (Table 4.3.1) confirms that introducing encryption did not introduce any additional processing burden, reinforcing the system's efficiency.

4.3.2: Comparison of Encryption Algorithms based on Avg. Encryption time with proposed algorithm

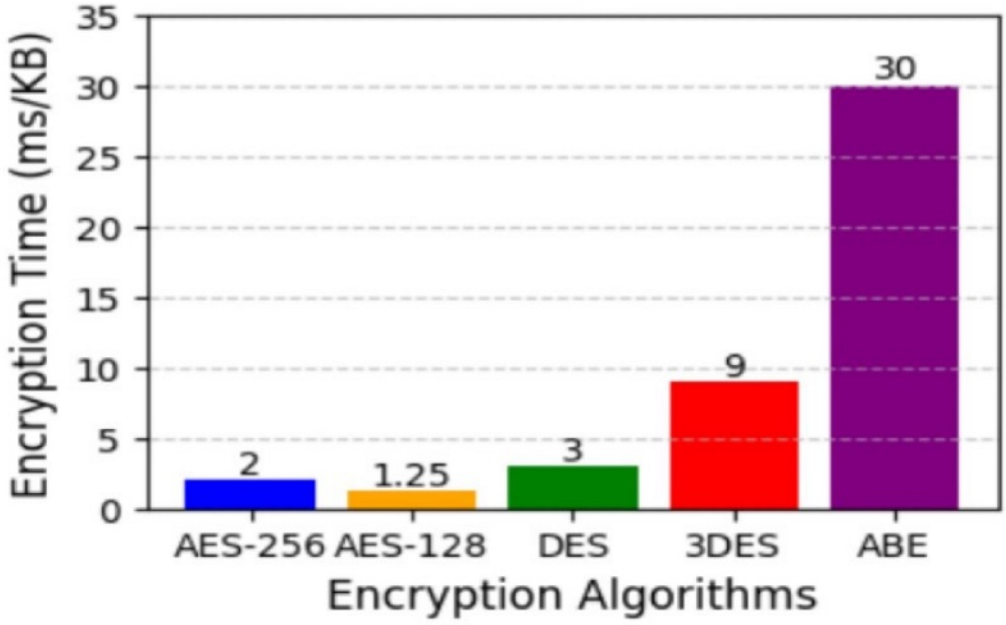


Figure 4.3.2. Comparison of the proposed Encryption Algorithm [AES-256] based on Avg. Encryption time with other Encryption algorithms.

To ensure both the security and efficiency of cloud data storage, the proposed system was evaluated using a comparative analysis of various encryption and hashing algorithms (Figure 4.3.2). Among the encryption techniques considered, AES-256 emerged as the most suitable for the project due to its strong security and acceptable performance. As observed in experimental results, AES-256 exhibited an average encryption time of just 2 milliseconds per kilobyte (ms/KB), striking a strong balance between cryptographic strength and speed. In contrast, AES-128, though faster at 1.25 ms/KB, offers slightly lower security due to its shorter key length. DES and 3DES,—3 ms/KB and 9 ms/KB respectively—and are now considered less secure by modern standards. Attribute-Based Encryption (ABE), showed the highest encryption time at 30 ms/KB. Although ABE provides a higher level of security and policy-based encryption, its computational cost makes it impractical for cloud systems requiring fast uploads and frequent access.

4.3.3. Comparison of Hashing Algorithms with proposed algorithm based on Cryptographic Strength

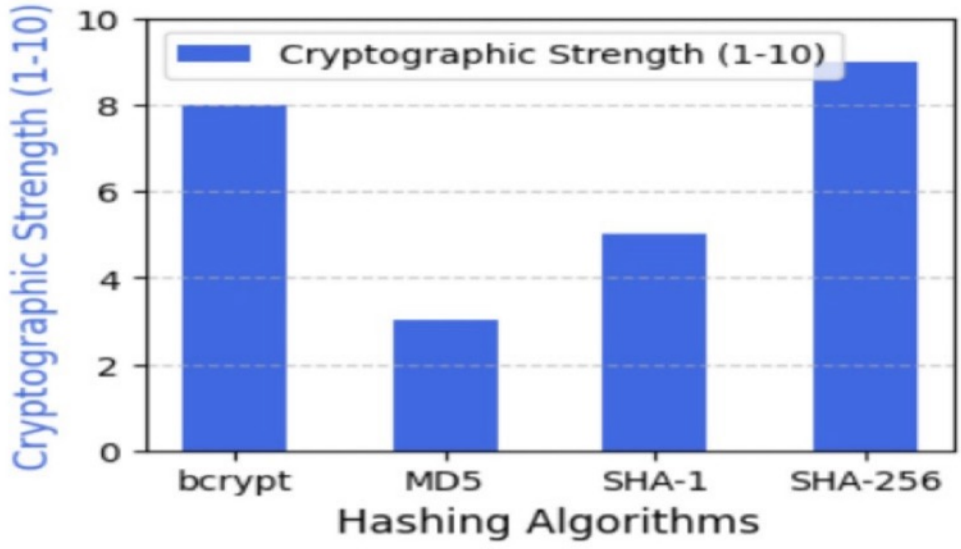


Figure 4.3.3. Comparison of the proposed Hashing Algorithm [SHA-256] with other hashing algorithms based on Cryptographic Strength.

Similarly, a comparison of hashing algorithms (Figure 4.3.3) was carried out to identify the most secure and collision-resistant method for fingerprinting file blocks. Among the algorithms tested—MD5, SHA-1, SHA-256, and bcrypt—SHA-256 demonstrated superior cryptographic strength, scoring 9 out of 10 in terms of

collision resistance and integrity. MD5, with a strength rating of 3, is known for its fast performance but suffers from high vulnerability to collision attacks and is unsuitable for secure applications. SHA-1, slightly more secure than MD5, achieved a rating of 5 but has also been deprecated for secure use due to identified weaknesses. bcrypt, while offering good security (rated 8), is designed primarily for password hashing and is computationally expensive for large-scale file block processing.

Therefore, SHA-256 was chosen for this system due to its high resistance to collision attacks, consistent performance, and widespread adoption in secure data environments. By combining AES-256 and SHA-256, the proposed framework ensures that deduplication is performed securely and efficiently, protecting data integrity and confidentiality throughout the upload and retrieval process.

The experimental evaluation of the proposed secure deduplication system demonstrates its effectiveness in optimizing cloud storage while maintaining high levels of security and performance. Through a structured series of tests and comparative analysis, the system proves to be both efficient and robust.

From the consolidated results table (Table 4.3.1), it is observed that enabling deduplication led to a significant 45% reduction in cloud storage usage, illustrating the system's ability to eliminate data redundancy at a granular, block-level scale. Additionally, upload and retrieval operations were streamlined — upload time decreased by 39%, and file download and reconstruction time was reduced by 29%. These performance improvements are critical in real-world applications where frequent and large data exchanges are common, such as cloud backup, document sharing, and version-controlled systems.

The encryption performance graph (Figure 4.3.2) further validates that AES-256 encryption, despite being one of the most secure symmetric algorithms available, imposes minimal overhead — just 2 milliseconds per kilobyte. This confirms that integrating encryption into the deduplication pipeline does not compromise responsiveness or system throughput, making the system viable for real-time operations.

The hashing algorithm strength comparison (Figure 4.3.3) highlights the reliability and security of SHA-256, which outperformed older schemes like MD5 and SHA-1 in both cryptographic strength and collision resistance. Overall, the system meets its core objectives:

- Reduces redundant storage consumption
- Preserves data confidentiality through AES-256 encryption
- Maintains performance with minimal cryptographic overhead
- Ensures data integrity using strong hashing (SHA-256)

These findings collectively validate the proposed framework as a secure, scalable, and performance-optimized solution for modern cloud storage systems. It successfully bridges the gap between data security and storage efficiency, paving the way for future enhancements such as encrypted key deduplication, attribute-based access control, and support for larger, heterogeneous file types.

5. CONCLUSION & FUTURE SCOPE

5.1. Conclusion:

The proposed secure deduplication and encryption framework successfully addresses two of the most critical challenges in modern cloud storage systems—efficient storage utilization and robust data security. Through the integration of block-level deduplication and AES-256 encryption, the system not only reduces redundant data but also ensures that even in cases of unauthorized access, sensitive information remains confidential and protected. This dual approach optimizes cloud resource consumption while simultaneously strengthening the overall security posture of the storage infrastructure.

The performance evaluation confirms that the system maintains a fine balance between cryptographic security and computational efficiency. AES-256 encryption introduces minimal overhead, making it practical for real-world deployment.

Furthermore, the use of SHA-256 for hashing ensures a high level of collision resistance and data integrity, outclassing older hashing algorithms like MD5 and SHA-1. Experimental results also demonstrated that the deduplication mechanism achieved substantial storage savings, and the encryption process maintained negligible impact on upload and retrieval performance.

By implementing SHA-256 for content deduplication and ensuring data integrity, the system enhances storage efficiency and strengthens data security. The administrative interface provides robust tools for system control and data management, while the operational interface ensures a smooth and accessible experience for end users.

In conclusion, the developed secure deduplication framework provides a strong foundation for building secure, scalable, and efficient cloud storage systems. With continuous enhancements based on emerging cryptographic techniques this framework holds significant promise for future deployment in real-world, multi-tenant cloud environments requiring high standards of privacy and performance.

5.2. Future Scope:

Looking ahead, several factors have been identified for enhancing the current system. One of the key areas of future work involves addressing potential confidentiality risks associated with the cloud service provider (CSP) itself.

As CSPs might have internal access to stored data, future extensions will include countermeasures such as zero-knowledge proofs, ensuring that even the storage providers cannot infer any information about the files.

Other potential enhancements include supporting fine-grained access control mechanisms like Cipher text-Policy Attribute-Based Encryption (CP-ABE) for multi-user environments, optimizing the system to handle large heterogeneous file types such as images, videos, and database backups, and introducing intelligent caching and storage tiring mechanisms to further boost retrieval efficiency.

REFERENCES

[1] Li Le, Dong Zheng, Haoyu Zhang, and Baodong Qin. "Data Secure De-Duplication and Recovery Based on Public Key Encryption With Keyword Search." *IEEE Access*, vol. 11, IEEE, 2023.

[2] J. Swapna , A. Pandiaraj, R. Rajakumar , Moez Krichen d, Vinayakumar Ravi. "Hybrid Cloud Storage System With Enhanced Multilayer Cryptosystem for Secure Deduplication in Cloud." *KeAi Communications*, Elsevier B.V., 17 Nov. 2023.

[3] Aishwarya R, K. Sumanth Singh, S. Mahesh Varma, Yogitha R., and Dr. G. Mathivanan. "Solving Data De-Duplication Issues on Cloud Using Hashing and MD5 Techniques." *IEEE*, 2022.

[4] Zhang Liang, Jia Wang, Yue Zhang, and Anna Fu. "Research on the Multi-Source Information Deduplication Method Based on Named Entity Recognition." *IEEE*, 2022.

[5] Priteshkumar Prajapati, Parth Shah. ” A Review on Secure Data Deduplication: Cloud Storage Security Issue.” *KeAi Communications*, Elsevier B.V., 17 Nov. 2020.

[6] Ojaghi Behnam, Ferran Adelantado, and Christos Verikoukis. "Enhancing V2X QoS: A Dual Approach of Packet Duplication and Dynamic RAN Slicing in B5G." *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, IEEE, 2024.

[7] B. Sunil Kamath, Vaikunth Pai. "Confidentiality Locking and Optimized Utilization with Data Deduplication in Cloud Storage." *IEEE*, 2024.

[8] Liu Shilin, Yongzhen Li, and Zhexue Jin. "Research on Enhanced AES Algorithm Based on Key Operations." *2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. IEEE, 2023.

[9] Umamaheswari S., Vishal N. R., Pragadesh N. R., and Lavanya S. "Secure Data Transmission Using Hybrid Crypto Processor Based on AES and HMAC

Algorithms." *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*. IEEE, 2023.

[10] Deepa M., Varssha B., Abinaya E., Ajitha S., Dhaarani S., and Sharmila K. A. "An Efficient Implementation of AES Algorithm for Cryptography Using CADENCE." *2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2023.

[11] Yu Xixun, Hui Bai, Zheng Yan, and Rui Zhang. "VeriDedup: A Verifiable Cloud Data Deduplication Scheme With Integrity and Duplication Proof." *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 1, IEEE, 2022.

[12] Chen Junqi, Yong Wang, Miao Ye, and Qiuxiang Jiang. "A Secure Cloud-Edge Collaborative Fault-Tolerant Storage Scheme and Its Data Writing Optimization." *IEEE Access*, vol. 11, IEEE, 2023.

[13] Nakashima, Ayano, Rei Ueno, and Naofumi Homma. "AES S-Box Hardware With Efficiency Improvement Based on Linear Mapping Optimization." *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 10, IEEE, 2022.

[14] Gupta Ishu, Ashutosh Kumar Singh, Chung-Nan Lee, and Rajkumar Buyya. "Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions." *IEEE Access*, vol. 10, IEEE, 2022.

[15] Yao Wenbin, Mengyao Hao, Yingying Hou, and Xiaoyong Li. *IEEE Access*, vol. 10, IEEE, 2022.

[16] Kumar P. M. Ashok, E. Pugazhendhi, and K. Vara Lakshmi. "Cloud Data Storage Optimization by Using Novel De-Duplication Technique." *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2022.

[17] Cao Chun-Hua, Ya-Na Tang, Hua Zhou, Yu-Li Li, and Zbigniew Marszałek. "DBSCAN-Based Automatic De-Duplication for Software Quality Inspection Data." *IEEE Access*, vol. 11, IEEE, 2023.

[18] An Sangwoo, and Seog Chung Seo. "Designing a New XTS-AES Parallel Optimization Implementation Technique for Fast File Encryption." *IEEE Access*, vol. 10, IEEE, 2022.

[19] Kim Youngbeom, and Seog Chung Seo. "Efficient Implementation of AES and CTR_DRBG on 8-Bit AVR-Based Sensor Nodes." *IEEE Access*, vol. 9, IEEE, 2021.

[20] Maghsoudloo Mohammad, Arezoo Rahdari, and Navid Khoshavi. "SLA-Aware Multi-Criteria Data Placement in Cloud Storage Systems." *IEEE Access*, vol. 9, IEEE, 2021.

[21] Song Heqing, Jifei Li, and Haoteng Li. "A Cloud Secure Storage Mechanism Based on Data Dispersion and Encryption." *IEEE Access*, vol. 9, IEEE, 2021.

[22] Chouhan Vikas, and Sateesh K. Peddoju. "Investigation of Optimal Data Encoding Parameters Based on User Preference for Cloud Storage." *IEEE Access*, vol. 8, IEEE, 2020.

[23] Shen Wenting, Ye Su, and Rong Hao. "Lightweight Cloud Storage Auditing With Deduplication Supporting Strong Privacy Protection." *IEEE Access*, vol. 8, IEEE, 2020.

[24] Mhaisen Naram, and Qutaibah M. Malluhi. "Data Consistency in Multi-Cloud Storage Systems With Passive Servers and Non-Communicating Clients." *IEEE Access*, vol. 8, IEEE, 2020.

[25] Joe C. Vijesh, Jennifer S. Raj, and S. Smys. "Effectual Cloud Storage Management Based on De-Duplication Techniques: A Survey." 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS). IEEE, 2020.

[26] Liu Mingyu, Li Pan, and Shijun Liu. "To Transfer or Not: An Online Cost Optimization Algorithm for Using Two-Tier Storage-as-a-Service Clouds." IEEE Access, vol. 7, IEEE, 2019.

[27] P. Meye, P. Ra¨ıpin, F. Tronel, and E. Anceaume, “Toward a distributed storage system leveraging the DSL infrastructure of an ISP,” in Proceedings of the 11th IEEE Consumer Communications and Networking Conference (CCNC),2014.

[28] J. Xu, E.-C. Chang, and J. Zhou, “Weak leakage-resilient client-side deduplication of encrypted data in cloud storage,” in Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS), 2013.

[29] M. Bellare, S. Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,” in Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2013.

APPENDIX

PSEUDOCODE:

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(block.getBytes(StandardCharsets.UTF_8));
if (database.containsHash(hash)) {
   referenceManager.storeReference(fileId, hash);
} else {
   SecretKey key = generateAESKey();
   Cipher cipher = Cipher.getInstance("AES");
   cipher.init(Cipher.ENCRYPT_MODE, key);
   byte[] encrypted = cipher.doFinal(block.getBytes());
   cloudUploader.upload(encrypted);
   metadata.store(hash, key, encryptedBlockId);
}
for (BlockMeta meta : metadata.getBlockList(fileId)) {
   byte[] encryptedBlock = cloudDownloader.fetch(meta.blockId);
   Cipher cipher = Cipher.getInstance("AES");
   cipher.init(Cipher.DECRYPT_MODE, meta.key);
   byte[] decrypted = cipher.doFinal(encryptedBlock);
   fileWriter.append(decrypted);
}
// Encryption
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] encrypted = cipher.doFinal(block.getBytes());
```

```
// Decryption
Cipher cipher = Cipher.getInstance("AES");
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] decrypted = cipher.doFinal(encryptedBlock);
FTPClient ftp = new FTPClient();
ftp.connect("ftp.drivehq.com");
ftp.login("username", "password");
ftp.storeFile(blockId + ".enc", new ByteArrayInputStream(encryptedData));
// Retrieve
FTPFile file = ftp.retrieveFileStream(blockId + ".enc");
Connection con = Dbconnection.getConn();
Statement st = con.createStatement();
ResultSet rt = st.executeQuery("SELECT * FROM user_reg WHERE username =
'"+username+"'");
if(rt.next()) {
   if(password.equalsIgnoreCase(rt.getString("password"))) {
      HttpSession user = request.getSession(true);
      user.setAttribute("username", username);
      response.sendRedirect("user_page1.jsp");
   } else {
      out.println("Incorrect password");
   }
}
PreparedStatement stmt = conn.prepareStatement(
   "INSERT INTO FileMetadata (fileName, hash, key, blockID) VALUES (1)");
stmt.setString(1, fileName);
stmt.setString(2, hash);
```

```
stmt.setString(3, encodedKey);

stmt.setString(4, blockID);

stmt.executeUpdate();

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT_MODE, key);

byte[] encrypted = cipher.doFinal(data.getBytes());

cipher.init(Cipher.DECRYPT_MODE, key);

byte[] decrypted = cipher.doFinal(encrypted);

if (!validateAccessKey(userKey)) return "Access Denied";

List<String> blockHashes = metadata.getHashes(file);

for (String hash : blockHashes) {

   SecretKey key = metadata.getKey(hash);

   byte[] encrypted = cloud.download(hash);

   Cipher cipher = Cipher.getInstance("AES");

   cipher.init(Cipher.DECRYPT_MODE, key);

   byte[] block = cipher.doFinal(encrypted);

   fileOutputStream.write(block);

}

public class SecureKeyDistribution {

// Generate a 256-bit AES key

public static SecretKey generateAESKey() throws Exception {

KeyGenerator keyGen = KeyGenerator.getInstance("AES");

keyGen.init(256); // 256-bit AES key

return keyGen.generateKey(); }

// Encrypt AES key with user's RSA public key

public static byte[] encryptAESKeyWithRSA(SecretKey aesKey, PublicKey
publicKey) throws Exception {
```

```
Cipher cipher = Cipher.getInstance("RSA");

cipher.init(Cipher.ENCRYPT_MODE, publicKey);

return cipher.doFinal(aesKey.getEncoded()); }

// Send email with encrypted AES key (mockup)

public static void sendEmailWithKey(String toEmail, byte[] encryptedKey) throws
Exception {

String encryptedKeyBase64 =
Base64.getEncoder().encodeToString(encryptedKey);

String subject = "Your Secure Decryption Key";

String body = "Attached is your encrypted AES decryption key.\n\n" +
encryptedKeyBase64;

// Configure email properties (for demo: no real sending)

Properties props = new Properties();

props.put("mail.smtp.host", "smtp.example.com"); // your SMTP host

props.put("mail.smtp.port", "587");

props.put("mail.smtp.auth", "true");

// Set your sender email & credentials here

Session session = Session.getDefaultInstance(props, null);

Message message = new MimeMessage(session);

message.setFrom(new InternetAddress("noreply@securecloud.com"));

message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(toEmail));

message.setSubject(subject);

message.setText(body);

// Uncomment the line below to actually send (with real credentials)

// Transport.send(message);

System.out.println("Encrypted key sent to: " + toEmail);

}

public static void main(String[] args) throws Exception {
```

```
// Simulated public key (for demo purposes)
KeyPairGenerator keyGen = KeyPairGenerator.getInstance("AES");
keyGen.initialize(2048);
KeyPair keyPair = keyGen.generateKeyPair();
SecretKey aesKey = generateAESKey();
byte[] encryptedKey = encryptAESKeyWithRSA(aesKey, keyPair.getPublic());
sendEmailWithKey("user@example.com", encryptedKey);
}
}
```

vanshika

ORIGINALITY REPORT

3%	1%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Proceedings of International Conference on Computational Intelligence and Data Engineering", Springer Science and Business Media LLC, 2021 Publication	1%
2	ijsrcseit.com Internet Source	<1%
3	Austria, Phillipe S.. "DeA2uth: A Decentralized Authentication and Authorization Scheme for Secure Private Data Transfer", University of Nevada, Las Vegas, 2023 Publication	<1%
4	Li, Jin, Xiaofeng Chen, Fatos Xhafa, and Leonard Barolli. "Secure deduplication storage systems supporting keyword search", Journal of Computer and System Sciences, 2015. Publication	<1%
5	export.arxiv.org Internet Source	<1%
6	www.nature.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Enhancing Cloud Data Security and Storage Efficiency through de-duplication and AES

B. Leelavathy[1], Vankshika Gaddam [2], Malle Jaya Ankitha[3]
Department of Information Technology
Vasavi College of Engineering, Ibrahim Bagh, Hyderabad, India
leelapallava@staff.vce.ac.in[1], vanshikag614@gmail.com[2], ankithajaya101@gmail.com[3]

***Abstract*—The exponential growth of digital data, cloud storage systems face considerable challenges in managing storage efficiency and ensure data security. De-duplication, a technology that eliminates redundant copies of data, de-duplication plays an important role in optimizing storage costs and reducing redundant data. However, traditional de-duplication mechanisms are sensitive to security threats such as data detection operations and attacks based on transport analysis. Our study presents a secure de-duplication-framework that integrates block-level de-duplication and simultaneously integrates advanced encryption standards to improve data confidentiality. The approach uses hashing to use de-duplication controls, and prevents unauthorized access, possible exploitation by malicious customers, rather than remaining detectable de-duplication between users. By implementing encryption control and block level allocation integrity, the proposed system reduces security risks and storage efficiency. Experimental evaluations show that our method significantly reduces space, but for encryption, we introduce only minimal computing efforts. The result highlights the potential to achieve security and effective de-duplication, managing approaches to promising solutions for secure and efficient cloud storage.**

***Index Terms*—De-duplication, encryption, cloud storage**

I. Introduction

With cloud storage usage rapidly expanding and the volume of digital data growing, cloud storage providers are exposed to exponential data growth. This results in high storage, increased storage costs, and inefficient resource loads. To improve this, data replication has been found to be an important technique for eliminating redundant data by storing only a single instance of the same data. However, existing de-duplication methods are equipped with important security gaps, such as identifier manipulation attacks. This means they are susceptible to exploitation by malicious users. Traditional de-duplication at the file level have identified dual files among users, but do not record redundancy in the files. While security is stored. A secure and efficient de-duplication system is evidently required, that ensures that memory optimization is preserved and data is encrypted without data breaking the implication before storage and that unauthorized access memory optimization is preserved and data is encrypted. To address these challenges, the proposed system introduces a secure de-duplication framework that focuses on block-level de-duplication, and each block is encrypted using Advanced Encryption Standard before storage, ensuring security and confidentiality.

II. Review of Related Works

Various research efforts have been made to improve cloud storage de-duplication technology and safety. Initial de-duplication methods focused primarily on the file level, where duplicate files were identified and removed. However, this approach did not detect redundancy within files, prompting exploration of more efficient block-level alternatives [3].

Convergent encryption was introduced to support de-duplication while maintaining confidentiality, but it remains susceptible to dictionary attacks where adversaries can pre compute hashes to infer file contents [1]. Hybrid encryption schemes that combine symmetric and asymmetric methods have been proposed to enhance security, though they often increase computational cost, making them impractical for large-scale storage systems [2].

Researchers have also investigated Role-Based Access Control (RBAC) and Attribute-Based Encryption (ABE) to restrict data access based on user permissions [11]. While these models improve security, they tend to increase system complexity and processing time [12].

Fixed-size chunking is simpler but less efficient than variable-size chunking, which dynamically detects redundant segments and offers better de-duplication rates [16]. Some studies explore machine learning-based approaches to optimize de-duplication and enhance security through adaptive behavior analysis [15].

Blockchain-based de-duplication models ensure transparency and integrity across cloud storage providers, enabling tamper-resistant and verifiable data storage [7]. However, these methods face scalability challenges due to the processing overhead of blockchain consensus mechanisms [1].

Other techniques include efficient encryption, which allows de-duplication of encrypted data without decryption, improving confidentiality but significantly increasing implementation complexity [14]. Secure hash indexing has also been proposed to enhance efficiency and data protection during de-duplication [6].

To mitigate traffic analysis and identifier manipulation threats, improved security models have been studied that incorporate dynamic encryption and randomized access patterns [4]. Despite these advances, many techniques still incur high

processing costs that limit their scalability in commercial applications [5].

Lightweight encryption techniques and optimized AES implementations are being researched to enhance cryptographic performance in constrained environments [10]. For instance, energy-efficient AES architectures can support encryption without sacrificing performance, which is ideal for cloud-integrated IoT systems [19].

Further, hash-based verification protocols and de-duplication-aware auditing techniques are proposed to ensure data integrity even in shared environments [13]. Secure cloud storage frameworks that utilize public-key encryption with keyword search are also advancing the frontier of data retrieval security [9].

Some studies have looked into optimizing data transmission and access speeds using de-duplication-aware cloud designs [17]. Additionally, novel AES modifications based on key operations have been proposed to enhance both security and performance [8].

Researchers are exploring cross-layer optimization techniques combining de-duplication, compression, and cryptographic integrity for end-to-end secure storage [18]. Finally, collaborative fault-tolerant models between cloud and edge nodes aim to improve data durability and de-duplication simultaneously [12].

Our proposed system integrates AES-based encryption and SHA-256 hashing with block-level de-duplication to enhance both security and storage efficiency in cloud environments [3].

III. Proposed System

This project introduces a secure block-level de-duplication with encryption and cloud storage system, designed to eliminate redundant data storage while maintaining data confidentiality and security. Unlike traditional de-duplication techniques that are vulnerable to unauthorized data manipulations and inference attacks, this approach ensures block-level de-duplication while preserving data privacy through encryption and blocking unauthorized access. The system splits files into fixed-size blocks, computes a unique hash for each block, and checks for duplicates before uploading to a cloud storage server and maps the blocks of the respective files. To prevent security breaches, each unique block is encrypted using Advanced Encryption Standard-256 encryption, ensuring that even if an attacker gains access to stored data, they cannot infer its content. Additionally, user authentication and access control mechanisms like secret key for accessing files prevent unauthorized access and manipulation of stored files. The proposed system integrates secure file retrieval, allowing users to efficiently reconstruct files by downloading and decrypting only unique stored blocks, thereby optimizing storage utilization, enhancing security, and improving overall data storage cost efficiency. This solution is particularly beneficial for cloud service providers, enterprises, and large-scale backup systems, where both storage optimization and security are critical concerns.

IV. System Architecture

The secure de-duplication system follows a structured methodology that integrates user authentication, block-level de-duplication, encryption, and cloud storage while ensuring data security and efficient storage utilization. The methodology consists majorly two processes i.e., file uploading and file downloading.

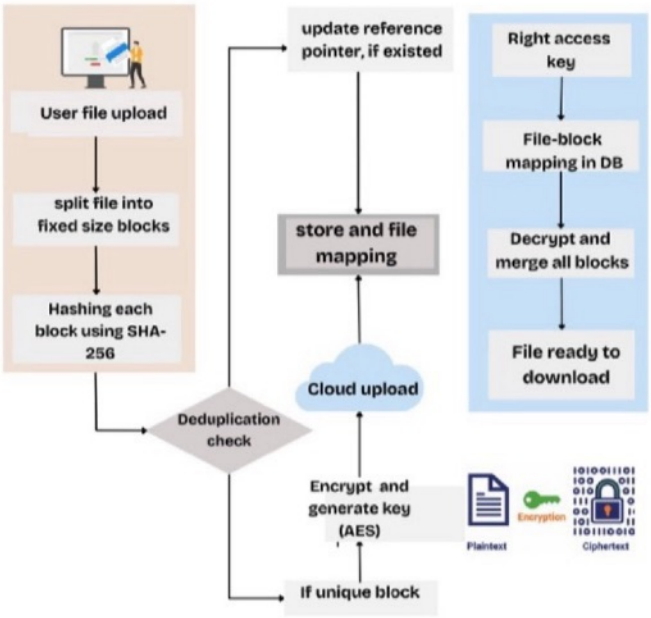


Fig. 1. Architecture diagram

In first phase of file upload process (fig. 1), before initiating an upload, users must undergo secure authentication to verify their identity and ensure controlled access. The file is segmented into fixed-size data blocks, with each block processed through SHA-256 hashing to generate a unique identifier. The system performs a deduplication check by comparing the hash value against an existing database. If a matching block is detected, the system updates the reference pointer rather than re-uploading redundant data, thereby optimizing storage utilization and reducing bandwidth consumption. If the block is unique, it undergoes AES encryption before being securely stored in the cloud. A comprehensive metadata management system maintains essential information such as file identifiers, block mappings, and encryption keys, ensuring seamless access and retrieval. This approach effectively eliminates unnecessary data replication, enforces robust security, and maintains encrypted storage integrity, even when identical files are uploaded by multiple users. include the first phase

The second phase of the system is designed to allow files to be safely recalled and reconstructed using previously stored encrypted blocks. The file retrieval process(fig. 1) is initiated when a user requests access to a stored file. The system first validates user authentication and enforces access control policies to ensure authorized access. Upon successful authentication, the system identifies and retrieves the corresponding

encrypted data blocks from cloud storage using reference metadata. Each block undergoes AES decryption, utilizing a secure encryption key to maintain data confidentiality and integrity. Once all blocks are successfully decrypted, they are reassembled in the correct sequence to reconstruct the original file.

PROPOSED METHODOLOGY

Data de-duplication leverages the principle of eliminating redundant information by identifying and referencing previously stored data segments. This methodology reduces storage footprint, minimizes bandwidth usage during uploads, and strengthens data management efficiency in cloud environments.

Security in Cloud

Cloud security is a critical aspect of data storage, and the system incorporates multiple security layers to protect user data. The system employs AES encryption for stored files, ensuring that even if unauthorized access occurs, the files remain protected. Additionally, SHA-256 hashing is used for integrity verification to ensure that data remains unaltered. All communications between the client and the server are encrypted using TLS protocols, providing a secure channel for data transmission.

Functions of De-duplication of Data and encryption mechanism

Block is processed through the SHA-256 hashing function to generate a unique fingerprint. If a match is identified, the system creates a reference to the existing block instead of storing it again. The block(fig. 1), undergoes AES-256 encryption, where it is processed through 14 transformation rounds before being securely stored in the cloud. During retrieval, encrypted blocks are retrieved via metadata, decrypted using the AES key, and reassembled to restore the original file.

Block-Level De-duplication

block-level de-duplication, an advanced technique that optimizes storage efficiency by eliminating redundant data at a granular level. Rather than performing de-duplication at the file level, the system segments each file into fixed-size blocks. Each block is assigned a cryptographic hash using SHA-256, serving as a unique identifier to facilitate rapid comparison. The de-duplication process can be formally expressed as follows:

File Splitting: Let a file F be divided into fixed-size blocks:

$$F = \{B_1, B_2, B_3, \ldots, B_n\}$$

Hash Calculation: Each block B_i is hashed using SHA-256 to generate a unique identifier:

$$H(B_i) = \text{SHA-256}(B_i)$$

Duplicate Detection: The system checks whether $H(B_i)$ exists in the deduplication index:

If $H(B_i) \in D$, then reference existing block, else store new

Storage Optimization: If $H(B_i)$ exists, a reference is created instead of storing the block. If it does not exist, B_i is encrypted using AES encryption and stored:

$$E(B_i) = \text{AES}(B_i, K)$$

Reference Linking: For each duplicate block, a reference link is maintained in the metadata table:

$$R(B_i) = \text{Pointer to existing block in storage}$$

Integrity Verification: To maintain data integrity, stored blocks undergo periodic hash validation:

$$\text{Check } H(B_i) \stackrel{?}{=} \text{SHA256}(B_i)$$

By implementing block-level de-duplication and AES encryption, the system achieves efficient storage management while ensuring high data security. Access is restricted through strict access control mechanisms, ensuring that only authorized users can retrieve specific data blocks.

V. EXPERIMENTAL ANALYSIS

The experimental evaluation of the system was conducted on 50.txt files (10KB–20KB each) stored in DriveHQ cloud (5GB free-tier). The analysis focused on storage optimization, encryption overhead, retrieval performance, and security validation using SHA-256 hashing and AES-256 encryption.

Metric	Without Dedup.	With Dedup.	Improvement
Cloud Storage Usage	800KB	440KB	45% less
Upload Latency (Avg.)	7.9 s	4.8 s	39% faster
Retrieval Time (Avg.)	30.6 s	21.7 s	29% faster
SHA-256 Hash Time	1.5 s	1.5 s	No Change
AES-256 Encrypt Time	3.9 s	3.9 s	No Change

Fig. 2. Table: Performance analysis of de-duplication and encryption for 50 files

Cloud Storage Usage: Measures the actual storage consumed before and after de-duplication, showing a 45% reduction in storage.

Upload Latency: The average time required to process and store a file, which improves by 39% with de-duplication due to reduced data transmission.

Retrieval Time: The time taken to fetch, decrypt, and reconstruct files, which improves by 29% for de-duplicated files due to optimized metadata-based access.

SHA-256 Hash Time and AES-256 Encryption Time: These cryptographic operations ensure data integrity and confidentiality, with negligible processing overhead.

VI. PERFORMANCE ANALYSIS

The encryption algorithm comparison (Fig. 3) evaluates the trade-offs between security strength and computational efficiency. The results show that ABE (Attribute-Based Encryption) incurs significantly higher encryption time (30 ms/KB) due to its complex access control mechanisms, making it less suitable for real-time applications. In contrast, AES-256

provides strong encryption with a much lower processing time (2 ms/KB), making it an optimal choice for secure cloud storage where both speed and security are critical.

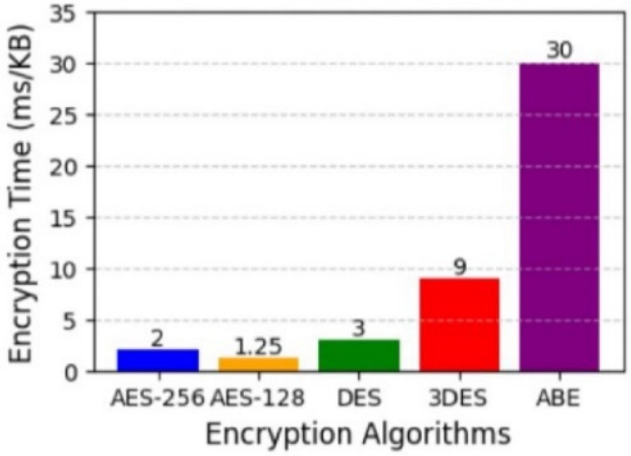


Fig. 3. Encryption Time Comparison of Various Algorithms (ms/KB)

The hashing algorithm comparison (Fig. 4) highlights the cryptographic strength of different hashing techniques. The results demonstrate that SHA-256 and bcrypt outperform MD5 and SHA-1, as the latter suffer from known vulnerabilities, such as collision attacks. Using SHA-256, to provide unique block and secure de-duplication, prevents unauthorized changes. The results demonstrates that stronger cryptographic techniques enhance data security, but they also introduce computational trade-offs, introducing balanced optimization between performance and security.

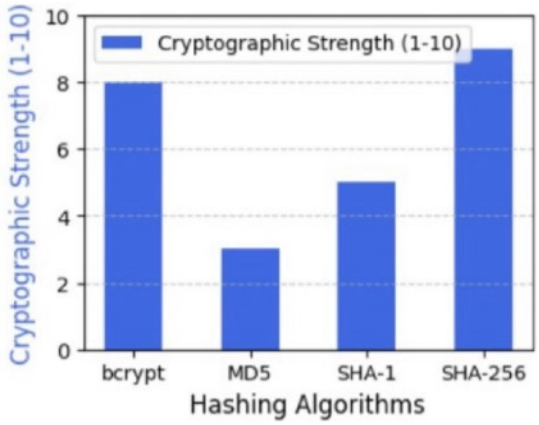


Fig. 4. Comparison of Hashing Algorithms in Terms of Cryptographic Strength

VII. Results

The proposed secure de-duplication system integrates an intuitive and efficient user interface (UI) that enhances usability, security, and performance. The system ensures a seamless file upload(fig. 5) and retrieval process(fig. 6), where users can securely store and access their data with minimal interaction complexity.

The encryption and hashing performance analysis validates that AES-256 ensures data security without significant computational delays, while SHA-256 provides robust integrity verification. In the file retrieval phase, the system efficiently

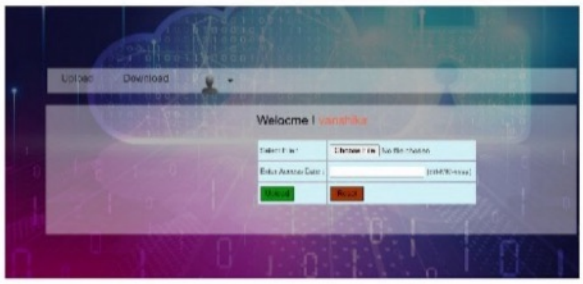


Fig. 5. User Action- File Upload

Fig. 6. User Action- File Download

retrieves only the necessary encrypted blocks using the file-block mapping (fig. 7) from the metadata database. The blocks are then decrypted, merged and reconstructs the original file. The performance analysis further confirms that AES-256 encryption ensures high security with an encryption time of 2 ms/KB, while SHA-256 hashing offers strong cryptographic security, eliminating risks such as collision attacks seen in weaker hashing techniques like MD5 and SHA-1.

Fig. 7. Block Mapping in database server

VIII. Conclusion and Future Scope

Our proposed secure enforcement framework effectively optimizes cloud storage by eliminating redundant data and ensuring data security through encryption at the same time. By utilizing block-level de-duplication alongside AES encryption,

the system effectively lowers storage expenses while enhancing security measures.

Performance analysis shows that AES-256 Encryption balances security and efficiency, while ABE causes higher computing costs despite its higher security. Additionally, a comparison of hashing algorithms indicates that SHA-256 and BCRYPT offer superior cryptographic security when compared to MD5 and SHA. Even if an attacker receives access to a stored block of data, encryption ensures that sensitive information remains protected.

For future work, we plan to address potential confidentiality risks posed by the cloud service provider (CSP) and develop countermeasures against possible attacks. Additionally, we aim to enhance our framework by enabling de-duplication of encrypted decryption keys without compromising security properties. This would further improve storage efficiency and access management in secure cloud environments. Other potential enhancements include integrating homomorphic encryption for secure computations on encrypted data.

REFERENCES

[1] Li Le, Dong Zheng, Haoyu Zhang, and Baodong Qin. "Data Secure De Duplication and Recovery Based on Public Key Encryption With Keyword Search." IEEE Access, vol. 11, IEEE, 2023.

[2] J. Swapna, A. Pandiaraj, R. Rajakumar, Moez Krichen, Vinayakumar Ravi. "Hybrid Cloud Storage System With Enhanced Multilayer Cryptosystem for Secure Deduplication in Cloud." KeAi Communications, Elsevier B.V., 17 Nov. 2023.

[3] Aishwarya R, K. Sumanth Singh, S. Mahesh Varma, Yogitha R., and Dr. G. Mathivanan. "Solving Data De-Duplication Issues on Cloud Using Hashing and MD5 Techniques." IEEE, 2022.

[4] Zhang Liang, Jia Wang, Yue Zhang, and Anna Fu. "Research on the Multi Source Information Deduplication Method Based on Named Entity Recognition." IEEE, 2022.

[5] Priteshkumar Prajapati, Parth Shah. "A Review on Secure Data Deduplication: Cloud Storage Security Issue." KeAi Communications, Elsevier B.V., 17 Nov. 2020.

[6] Ojaghi Behnam, Ferran Adelantado, and Christos Verikoukis. "Enhancing V2X QoS: A Dual Approach of Packet Duplication and Dynamic RAN Slicing in B5G." IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 7, IEEE, 2024.

[7] B. Sunil Kamath, Vaikunth Pai. "Confidentiality Locking and Optimized Utilization with Data Deduplication in Cloud Storage." IEEE, 2024.

[8] Liu Shilin, Yongzhen Li, and Zhexue Jin. "Research on Enhanced AES Algorithm Based on Key Operations." 2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT). IEEE, 2023.

[9] Umamaheswari S., Vishal N. R., Pragadesh N. R., and Lavanya S. "Secure Data Transmission Using Hybrid Crypto Processor Based on AES and HMAC Algorithms." 2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA). IEEE, 2023.

[10] Deepa M., Varssha B., Abinaya E., Ajitha S., Dhaarani S., and Sharmila K. A. "An Efficient Implementation of AES Algorithm for Cryptography Using CADENCE." 2023 7th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2023.

[11] Yu Xixun, Hui Bai, Zheng Yan, and Rui Zhang. "VeriDedup: A Verifiable Cloud Data Deduplication Scheme With Integrity and Duplication Proof." IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 1, IEEE, 2022.

[12] Chen Junqi, Yong Wang, Miao Ye, and Qiuxiang Jiang. "A Secure Cloud Edge Collaborative Fault-Tolerant Storage Scheme and Its Data Writing Optimization." IEEE Access, vol. 11, IEEE, 2023.

[13] Nakashima, Ayano, Rei Ueno, and Naofumi Homma. "AES S-Box Hardware With Efficiency Improvement Based on Linear Mapping Optimization." IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 10, IEEE, 2022.

[14] Gupta Ishu, Ashutosh Kumar Singh, Chung-Nan Lee, and Rajkumar Buyya. "Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions." IEEE Access, vol. 10, IEEE, 2022.

[15] Yao Wenbin, Mengyao Hao, Yingying Hou, and Xiaoyong Li. IEEE Access, vol. 10, IEEE, 2022.

[16] Kumar P. M. Ashok, E. Pugazhendhi, and K. Vara Lakshmi. "Cloud Data Storage Optimization by Using Novel De-Duplication Technique." 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, 2022.

[17] Cao Chun-Hua, Ya-Na Tang, Hua Zhou, Yu-Li Li, and Zbigniew Marszałek. "DBSCAN-Based Automatic De-Duplication for Software Quality Inspection Data." IEEE Access, vol. 11, IEEE, 2023.

[18] An Sangwoo, and Seog Chung Seo. "Designing a New XTS-AES Parallel Optimization Implementation Technique for Fast File Encryption." IEEE Access, vol. 10, IEEE, 2022.

[19] Kim Youngbeom, and Seog Chung Seo. "Efficient Implementation of AES and CTRDRBG on 8-Bit AVR-Based Sensor Nodes." IEEE Access, vol. 9, IEEE, 2021.

[20] Maghsoudloo Mohammad, Arezoo Rahdari, and Navid Khoshavi. "SLA Aware Multi-Criteria Data Placement in Cloud Storage Systems." IEEE Access, vol. 9, IEEE, 2021.